Optimising FOIL by new scoring functions[⋆]

P. Jiménez, J.L. Arjona, and J.L. Álvarez

University of Huelva, Department of Information Technology,
Escuela Politécnica Superior. Crta. Huelva - La Rábida. Palos de la Frontera 21071
patricia.jimenez@dti.uhu.es, arjona@dti.uhu.es, alvarez@uhu.es,
WWW home page: http://www.tdg-seville.info/

Abstract. FOIL is an Inductive Logic Programming Algorithm to discover first order rules to explain the patterns involved in a domain of knowledge. Domains with a huge amount of information are handicaps for FOIL due to the explosion of the search of space to devise the rules. Current solutions to problems in these domains are restricted to devising ad hoc domain dependent inductive algorithms that use a less-expressive formalism to code rules.
We work on optimising FOIL learning process to deal with such complex domain problems while retaining expressiveness. Our hypothesis is that changing the Information Gain scoring function, used by FOIL to decide how rules are learnt, can reduce the number of steps the algorithm performs. We have analysed 15 scoring functions, normalised them into a common notation and checked a test in which they are computed. The learning process will be evaluated according to its efficiency, and the quality of the rules according to their precision, recall, complexity and specificity. The results reinforce our hypothesis, demonstrating that replacing the Information Gain can optimise both the FOIL algorithm execution and the learnt rules.

Keywords: ILP, FOIL, scoring functions

1 Introduction

Machine learning systems aim to automatically learn to recognize complex patterns based on data from some background knowledge and to make intelligent decisions on new data. Many of these systems have their focus on Inductive Logic Programming (ILP), a subfield of machine learning which investigates the construction of first-order logic rules. This kind of systems include FOIL [19], GOLEM [12], PROGOL [11] with Shapiro's program MIS as one of their early predecessors [10].

However, a major problem of ILP systems arises when the set of training data is too large. The process of learning the hypothesis that best fits the available

[⋆] Supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, and TIN2008-04718-E).

knowledge, becomes inefficient or ineffective. This weakness is due to the explosion of the search space caused by the large number of rules it needs to evaluate. In domains like Information Extraction, it becomes an intractable task as was stated in SRV system [27]. An alternative choice is propositional logic systems, which have demonstrate their worth in real world applications, although they produce rules quite less expressive and consequently, they are restricted to be applied to simpler domains problems.

We wish to use FOIL algorithm to deal with such complex real-world domains. To apply FOIL to any domain efficiently, we suggest using different scoring functions instead of the Information Gain, which is employed by the original FOIL to select the best candidate rules. These scoring functions come from statistics, machine learning, and data mining literature. We wish to prove that using some of these scoring functions may find out best rules or find them out faster.

We bet on FOIL algorithm because of the amount of successors systems developed like FOCL [24], AUDREYII [17], mFOIL [16], HYDRA [15], FOSSIL [14], FFOIL [13], FZFOIL [26] and FOIDL [9]. They follow FOIL approach to devise the rules and have even tried to improve them. Some of these systems propose to use likelihood ratio, correlation criterion and estimated accuracy as alternatives to the Information Gain. In many cases, their results were better than FOIL ones but restricted to domain dependent tasks. So that the problem has not been solved yet. However, these systems and their results suggest that FOIL can be optimised in many ways.

The paper is organised as follows: first, we introduce an overview of FOIL algorithm and we propose to use new scoring functions in order to solve the mentioned problems. Next section a common notation and a set of scoring functions are explained. Then, we show a test where FOIL algorithm was applied and the results obtained. Conclusion section discuss these results and Future Research gives some tips to address our future work.

2 FOIL

In first order learning, training data comprises a target predicate, which is defined by a collection of positives and negatives examples according to whether they satisfy the target predicate or not. Therefore, a set of support predicates is defined either extensionally, similarly to what was previously made with the target predicate or intensionally, by means of a set of rules. The goal is to learn a set of logic rules that explain the target predicate in terms of itself and the support predicates.

FOIL is an algorithm of machine learning that induces first order rules. It uses separate-and-conquer method rather than divide-and conquer, focusing on creating a single rule at a time and removing any positive examples covered by each learnt rule. Then, it is invoked again to learn a second rule based on the remaining training examples. It is called a sequential covering algorithm because it sequentially learns a set of rules that together cover the full set of positive examples.

In order to learn each rule, it follows a top-down approach, starting with the most general header rule, and guided by a greedy search, is adding new literals to the rule, until it does not satisfy any negative example belonging to the target predicate. The set of rules is ready when none positive examples of the target predicate remain to be satisfied.

Each learnt rule is of the form $H \leftarrow B$ where H is the head and B is the body of the rule. H is a literal of the form $R(X_0, X_1, \ldots, X_n)$ where R is the target predicate and $X_0, X_1, \ldots, X_n$ are the variables. Similarly, B is a set of literals, for instance $P_1(X_0, X_2), P_2(X_3, X_1), \ldots$, where P_i represents any predicate defined in the knowledge base and $X_0, X_1, \ldots, X_n$ are the variables of the predicate P_i.

To add a new literal to the current rule, a list of candidate literals is generated. Each one is added to the current rule giving rise to a new candidate rule. The candidate rules are weighted based on Information Gain scoring function. It measures the benefit of replacing the current rule with a specific candidate rule. To compute this score, the Information Gain relies on the number of positive and negative examples that are satisfied before and after this replacement. The candidate rule with higher score is selected to keep growing.

Let tp be the number of positive examples and fp the number of negative examples that are satisfied by the current rule. The information conveyed by the knowledge that an example satisfied by the current rule is positive is given by

$$I(H \leftarrow B) = -log \frac{tp}{tp + fp} \tag{1}$$

Similarly, for each new candidate rule $I_k(H \leftarrow B')$, built from adding a new literal generated $P_k(X_1, .., X_n)$ to the current rule. Being t the number of positive examples satisfied by both the current and a new candidate rule, the Information Gain has a straightforward interpretation in terms of information theory and is given by the formula:

$$Gain(H \leftarrow B') = t \times (I_k(H \leftarrow B') - I(H \leftarrow B)) \tag{2}$$

In FZFOIL [26] some deficiencies in the Information Gain have been identified. Presumably, it may be due to the fact that Information Gain only take into account the number of positive and negative examples a new candidate rule satisfies, forgetting other parameters as the number of positive and negative examples this candidate rule discards.

For the purpose of improving the learning process, we analyse other scoring functions from the literature, trying to solve the Information Gain problem stated. They weigh up the candidate rules according to the existing correlation between each one and the current rule. Therefore, the gain of these scoring functions will measure the amount of correlation gained if the current rule is replaced with a new specific candidate rule.

3 Comparison Framework

The proposed scoring functions will be defined in terms of the well-known contingency table. For evaluating any first order candidate rule $H \leftarrow B'$, we rely on a contingency table as the one below in table 1.

		Predicted Class		
		R_k	$\neg R_k$	
Original Class	Positive	tp true positives	fn false negatives	
	Negative	fp false positives	tn true negatives	
				N

Table 1. Contingency Table

Original class represents all starting training examples both positive and negative (i.e., all training examples defined in the knowledge base for the target predicate and covered by the initial empty rule, which only has the header rule H). Predicted class are those examples satisfied by a specific candidate rule $R_k = H \leftarrow B'$ and those satisfied denying this rule (i.e., $\neg R_k = H \leftarrow \neg B'$). Thus, tp denotes the number of positive examples and fp denotes the number of negative examples that are satisfied by this candidate rule. Similarly, fn denotes the number of positive examples and tn denotes the number of negative examples discarded by this candidate rule. N is the total number of positive and negative training examples.

We have implemented and evaluated a subset of scoring functions proposed in [23] as objective measures and in [18] as measures for predictive and descriptive induction. Moreover, we have selected other scoring functions for being quite traditional. The set of scoring functions adapted to the previous contingency table is showed in table 4.

A summary description for each scoring function:

- Coverage is a measure of generality of a rule. If a rule characterizes more information in the data set, it tends to be more interesting.
- Laplace Accuracy [22] is an approximate measure to estimate the expected accuracy directly. General rules tend to be favored.
- Piatetski-Shapiro's [21] is one of the most frequently measure used in the evaluation of rules. It is also known as Leverage. It trades off generality and relative accuracy.
- $\phi-coefficient$ [20] is a statistical measure analogous to Pearson's product-moment correlation coefficient. It measures the degree of association between two binary variables (e.g., two rules, initial empty rule and a candidate rule). It is closely related to the χ^2 statistic since $\phi^2 = \frac{\chi^2}{N}$.
- Support [8] is a measure known from association rule learning, also called frequency. It is used for specifying if a rule is observed frequent enough in a data set.

- Rule Accuracy is also known as confidence [8]. It is related to the reliability. A rule is reliable if its predictions are highly accurate.
- Satisfaction [18] is similar to rule accuracy e.g., $Sat(H \leftarrow B) = 1$ if $RAcc(H \leftarrow B) = 1$, but, unlike Rule Accuracy, it takes the entire contingency table into account and is thus more suited towards knowledge discovery.
- Confirmation [7] is defined in terms of a modified χ^2 statistic. It trades off satisfaction and Leverage measures.
- F-measure is other statistic measure of a test accuracy. It considers both the precision and the recall of the test to compute the score. We compute F1-score which is the harmonic mean of precision and recall.
- kappa (κ) [6] captures the degree of agreement between a pair of variables (e.g., the initial empty rule and a candidate rule). If both variable are highly agree with each other, then the values for κ will result higher.
- Odds-ratio [4] represents the strength of association or non-independence between two binary data values. Unlike other measures of association for paired binary data, the comparison between the two variables is symmetrical.
- Yule's Q coefficient [5] is a normalized variant of the odds ratio.
- Lift (Interest) [2] is used quite extensively in data mining for measuring deviation from statistical independence. It gives an indication of rule significance or interest.
- Collective Strength [3] is other measure of correlation variant of Lift measure. It compares between actual and expected values. Higher values indicate perfectly positive correlation.
- Jaccard(ζ) [1] is a statistic used for comparing the similarity and diversity of sample sets. It is used extensively in information retrieval to measure the similarity between documents. We measure the similarity between two rules.

We have defined some measures to help us to decide which scoring function is more promising. The set of measures taking into account are:

1. ***Efficiency***. This is defined as the amount of useful work in relation to time and resources used. The resources are memory and space required.
2. ***Precision***, which measures the number of examples correctly satisfied by the set of rules against all examples satisfied, although so far, we only search for 100% accuracy rules, i.e., we do not allow rules that satisfy any negative example.
3. ***Recall***, which determines if a set of rules is complete, i.e., if it satisfies all positive examples belonging to the target predicate.
4. ***Complexity*** of the induced set of rules. It is computed in terms of bits from Minimum Description Length Principle [25].
5. ***Specificity/Generality*** of the induced set of rules. A rule is general if it satisfies most of positive examples belonging to the target predicate. It will be too specific when it only satisfies a few number of positive examples. Much more number of general rules, fewer rules in the set.

However, there are measures that can not be estimated objectively because they depend on other measures which we call secondary measures. For instance, secondary measures that may affect efficiency directly may be the number of backtracking performed or the number of candidate rules that were evaluated, which is one of the most expensive step in the algorithm. The number of different predicates used in the induced set of rules could also affect the efficiency of the learning process because it gives an idea about how well the knowledge base was built and therefore, how useful the support predicates defined are.

Secondary measures that may affect to generality/specificity of a rule are the number of variables used and the deep of the learnt rules, measured in terms of the number of literals in each rule. If these numbers are small it will mean that the rules are quite general, which is a desirable property.

Intuitively, the total number of induced rules will affect both, efficiency and generality/specificity. Fewer number of rules will make the process more efficient and the final set of rules more general. Note the former is true as long as the rule are not too complex and the latter is true as long as the set of rules has a good recall.

4 On Going Work

The example tested was first showed in [26]. Although it is a very simple problem with a negligible computational cost, it highlights the shortcomings of the Information Gain when selecting the best literal to add to the current rule. It tries to explain when somebody is sick. To carry out this task we rely on a set of seventeen individuals among which eight are sick, and the rest are not. The target predicate will be sick(X_0), which means the individual X_0 is sick. The support predicates defined to induce a set of rules that explain the target predicate sick(X_0) are:

- bearded(X_i), which means the individual X_i is bearded.
- smoker(X_i), which means the individual X_i is a smoker.
- father(X_i, X_j), which means X_i is X_j's father.
- boss(X_i, X_j), which means X_i is X_j's boss.

In the knowledge base, the positive examples belonging to the target predicate are all individuals who are sick. The rest are the negative examples and they can be defined explicitly or to be induced by Closed World Assumption. Similarly, we have to define the positive examples satisfied by each support predicate but there is no need to define the negative ones explicitly.

Our knowledge base would be a Prolog program as in table 5 and the set of learnt rules for this example is showed in table 3. The values obtained for the evaluation measures explained previously, are presented in table 2[1]

[1] where: **Prec.**: Precision, **C**: complexity (bits), **T**: time (milliseconds), **N**: number of candidates rules evaluated, **B**: number of backtracking, **L**: total number of unground literals, **V**: total number of variables, **P**: total number of different predicates, **S**: number of learnt rules.

Scoring Function	**Prec.**	**Recall**	**C**	**T**	**N**	**B**	**L**	**V**	**P**	**S**
Information Gain	1	1	23.75	6257	116	0	8	3	4	3
Coverage	1	0.5	12.17	138275	694	51	3	3	3	2
Laplace Accuracy	1	1	30.66	22317	408	0	13	6	6	3
Piatetski-Shapiro	1	1	22.34	5400	95	0	6	3	3	2
ϕ-coefficient	1	1	37.09	34200	183	1	9	4	3	3
Support	1	1	26.77	43759	191	8	7	4	5	2
Rule Accuracy	1	1	23.08	19017	375	0	11	6	5	3
Satisfaction	1	1	20.17	3512	68	0	6	2	5	3
Confirmation	1	1	20.17	3705	68	0	6	2	5	3
F-measure	0	0	0	58147	242	20	0	0	0	0
kappa (κ)	1	1	22.34	4983	95	0	6	3	3	2
Odds ratio	1	1	25.92	10470	158	3	8	3	4	3
Yule's Q	1	1	56.85	25701	267	4	14	4	5	6
Lift	1	1	23.08	17249	375	0	11	6	5	3
Collective Strength	1	1	22.34	4745	95	0	6	3	3	2
Jaccard(ζ)	0	0	0	57480	242	20	0	0	0	0

Table 2: table of results

5 Conclusions

As well as in other analysis of measures or scoring functions, we can not conclude saying there is a scoring function consistently better than the rest in all applications domains, although we have found some scoring functions that perform better than Information Gain in our running example.

The goal is to get the most reduced set of 100% accurate rules with the largest recall in the shortest time possible. To measure the quality of the rule, we need a balance among precision, recall, efficiency, complexity and specificity/generality to decide which is the most promising scoring function to achieve our goal but being the efficiency the most relevant factor. The ease of understanding the learnt rules is not an aspect to keep in mind because we do not yet who is going to interpret these output rules. It can be an expert user or a computer program.

Taking these factors into account, we consider that Collective Strength, Kappa and Piatetski-Shapiro's scoring function, maintaining the full recall, performed better. They evaluated only 95 candidate rules and the set of rules were more general (i.e., fewer number of learnt rules and fewer number of literals in the set of rules) and less complex in terms of bits. Satisfaction and Confirmation scoring functions are even better than the previous one. They evaluated fewer number of candidate rules and, although they had one rule more, the set of rules has a complexity still lower.

Support scoring function is quite similar to the Information Gain. It spent more time evaluating many candidate rules (i.e., a total number of 191 candidate rules were evaluated) and the rules are more complex but more general. As we consider efficiency the most relevant factor, we opt for Information Gain. However, Laplace accuracy, Odds-ratio, ϕ-coefficient, Rule Accuracy and Lift behaved worse than Information Gain, wasting time searching for more specific rules. Yule's Q is the worst scoring function within the group of functions that have full coverage.

Finally, Coverage, F-Measure and Jaccard scoring functions did not find out a set of rules with full coverage, they covers only a portion of positive examples defined in the knowledge base. The last two were unable to find out a single rule and they evaluated a huge amount of candidate rules caused, among other factors, by the backtracking performed. Note F-Measure and Jaccard scoring function has very similar formulae so they behaved in an identical way.

Despite of a single example is not reliable to get firm conclusions, it seems that scoring functions like Piatetski-Shapiro's, Confirmation, Satisfaction, Kappa and Collective Strength may be more promising than the Information Gain because they improve efficiency and quality of the rules obtained by the Information Gain, solving some shortcomings of this scoring function states. However, we need to perform an exhaustive comparative study and to establish a ranking among all of these scoring functions.

The application of this kind of systems is usually better than with any other known approach, so it needs to find more training sets to be tested, to define more additional measures if that would be necessary and to perform an exhaustive evaluation to get a reliable ranking. All this in order to apply FOIL satisfactorily to domains with a huge amount of information.

6 Future Research

We are working on collecting a set of training examples large enough to perform a reliable comparative study of these scoring functions and our own proposals of scoring functions. Furthermore, we intent to use the methodology explained in [23] to select the most fitting scoring function according to each problem.

Firstly, we will start with a set of training examples from Inductive Logic Programming and Association Rules domain to get an approximation of the behaviour of such scoring functions and compare our algorithm to other inductive logic programming algorithms, both recent and traditional. If we get improvements and its application is feasible in practical terms of efficiency, we shall collect new complex training examples especially from Information Extraction domain where currently, its application is intractable task both from the standpoint of computational cost as the user's point of view, who has to annotate each instance of an attribute (i.e., target predicate) that it wants to be extracted, which is a tedious and error-prone task.

Other proposals to improve FOIL algorithm also are considered. They are classified in optimisations and heuristics. Some optimisations consist of:

1. Parallelising the algorithm, i.e., guiding the search of each single rule in parallel avoiding overlapping amongst the rules being learnt. Overlap can be

avoided successfully by means of clustering or using the paired T-Students Test, which reduces the set of candidate rules into equivalence classes according to their similarity from statistical point of view.
2. Performing the exploration of the candidate rules incrementally, i.e., a single candidate rule is generated and checked. If it is good enough, the rest of candidate rules shall not be generated and checked, which shall reduce the entire search space. If the candidate rule being evaluated does not exceed the threshold of goodness defined, the next candidate rule shall be generated and checked, and so on until we find an acceptable candidate rule to add to the final set of learnt rules.Tuning this threshold of goodness is the key to take advantage of this optimisation.
3. Defining support predicates and/or semantic restrictions by means of a set of rules processed by a Prolog interpreter. The goal is to avoid the evaluation of certain candidate rules that present inconsistences amongst its predicates. For instance, since the algorithm makes a greedy search it can generate a rule like *title(X):-previous(X,Y), EQUAL(X,Y)*, which means that a piece of text is a title when it is preceded by itself, which obviously does not make sense. Furthermore, defining support predicates by means of a set of rules can also reduce the time spent by a user on building the knowledge base.
4. Translating output rules into regular expressions in order to improve the efficiency of the extraction process. Note that this optimisation is not generally applicable, but only to a subset of predicates on specific domains.
5. Incorporating fuzzification capabilities in the learning process of rules in order to relax the concept of inclusion or exclusion of a rule.

Next, we report on the heuristics we shall design. Note that the optimisations are intended to make inferring rules more efficient; heuristics, on the contrary, are designed to help guide the search process, i.e., in the worst case, they do not lead to an improvement.

1. Defining new criteria to rank the set of candidate rules; currently the top positions in the ranking are for those candidate rules that exceed a threshold, but if they do not exceed that threshold, the ranking is performed according to other properties as the number of new variables a rule contains, which are variables that are not present at the head of the rule, or properties related to their coverage. We think that alter the priority of these properties or taking, e.g., the distance to a goal rule, into account might help improve the search process of the rules.
2. When a rule becomes too complex, the system returns to a save point and restarts the search from it. A candidate rule is stored as a save point if its score is close to the score of the selected rule. We suggest defining new criteria to decide when a candidate rule can be stored as a save point. An alternative to the currently heuristic used is to store all candidate rules whose score exceeds Cantelli's threshold.

A Appendix

Scoring Function	**Rules**[2][3]
Information Gain	sick(X_0)$\leftarrow$ smoker(X_0). sick(X_0)$\leftarrow$ boss(X_1,X_0), sick(X_1), $\neg$father(X_0, X_2). sick(X_0)$\leftarrow$ boss(X_1,X_0), sick(X_1), father(X_2, X_0), sick(X_2).
Coverage	sick(X_0)$\leftarrow$ smoker(X_0). sick(X_0)$\leftarrow$ boss(X_0,X_1), father(X_0, X_2).
Laplace Accuracy	sick(X_0)$\leftarrow$ father(X_1,X_0), bearded(X_1), smoker(X_0). sick(X_0)$\leftarrow$ boss(X_1,X_0), sick(X_1), father(X_2,X_0), sick(X_2). sick(X_0)$\leftarrow$ $\neg$father(X_0,X_1), boss(X_1,X_2), boss(X_2,X_3), boss(X_4,X_1), father(X_5,X_2), $X_0 \neq X_3$.
Piatetski-Shapiro	sick(X_0)$\leftarrow$ father(X_1,X_0), bearded(X_1), boss(X_0,X_2). sick(X_0)$\leftarrow$ $\neg$father(X_0,X_1), boss(X_1,X_2), $\neg$boss(X_2,X_0).
ϕ-coefficient	sick(X_0)$\leftarrow$ $\neg$boss(X_0,X_1), boss(X_1,X_2), $\neg$boss(X_2,X_0). sick(X_0)$\leftarrow$ father(X_1,X_0), boss(X_0,X_2), $\neg$father(X_0,X_3). sick(X_0)$\leftarrow$ father(X_1,X_0), bearded(X_1), boss(X_0,X_2).
Support	sick(X_0)$\leftarrow$ boss(X_0,X_1), smoker(X_0). sick(X_0)$\leftarrow$ father(X_1,X_0), boss(X_1,X_2), $X_1 \neq X_2$, boss(X_1, X_3), sick(X_1).
Rule Accuracy	sick(X_0)$\leftarrow$ smoker(X_0). sick(X_0)$\leftarrow$ boss(X_1,X_0), sick(X_1), father(X_2,X_0), sick(X_2). sick(X_0)$\leftarrow$ $\neg$father(X_0,X_1), boss(X_1,X_2), boss(X_2,X_3), boss(X_4,X_1), father(X_5,X_2), $X_0 \neq X_3$.
Satisfaction	sick(X_0)$\leftarrow$ smoker(X_0). sick(X_0)$\leftarrow$ $\neg$father(X_0,X_1), $\neg$bearded(X_0). sick(X_0)$\leftarrow$ boss(X_1,X_0), sick(X_1), bearded(X_1).
Confirmation	sick(X_0)$\leftarrow$ smoker(X_0). sick(X_0)$\leftarrow$ $\neg$father(X_0,X_1), $\neg$bearded(X_0). sick(X_0)$\leftarrow$ boss(X_1, X_0), sick(X_1), bearded(X_1).
kappa (κ)	sick(X_0)$\leftarrow$ father(X_1,X_0), bearded(X_1), boss(X_0,X_2). sick(X_0)$\leftarrow$ $\neg$father(X_0,X_1), boss(X_1,X_2), $\neg$boss(X_2,X_0).
Odds-ratio	sick(X_0)$\leftarrow$ smoker(X_0). sick(X_0)$\leftarrow$ boss(X_1,X_0), boss(X_0,X_2), sick(X_2), sick(X_1). sick(X_0)$\leftarrow$ $\neg$boss(X_0,X_1), boss(X_1,X_2), $\neg$bearded(X_0).
Yule's Q	sick(X_0)$\leftarrow$ $\neg$bearded(X_0), boss(X_1,X_0), sick(X_1). sick(X_0)$\leftarrow$ boss(X_1,X_0), $\neg$father(X_1,X_2). sick(X_0)$\leftarrow$ smoker(X_0). sick(X_0)$\leftarrow$ father(X_1,X_0), $\neg$boss(X_2,X_0). sick(X_0)$\leftarrow$ boss(X_0,X_1), boss(X_2,X_0), smoker(X_1). sick(X_0)$\leftarrow$ boss(X_0,X_1), boss(X_2,X_0), $\neg$father(X_0,X_3).

[2] Note that FOIL relies on predefined predicates which are of the form $X_i = X_j$ or $X_i = c_j$, where X_i and X_j are variables and c_j is a constant (e.g., c_j can be any specific individual).

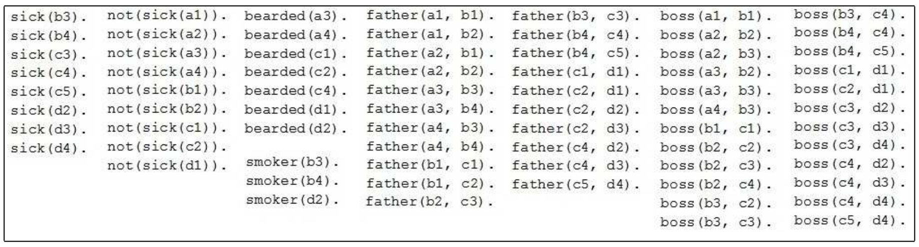
[3] Note that F-Measure and Jaccard scoring function did not get a set of rules.

Lift (Interest)	sick(X_0)← smoker(X_0). sick(X_0)← boss(X_1,X_0), sick(X_1), father(X_2,X_0), sick(X_2). sick(X_0)← ¬father(X_0,X_1), boss(X_1,X_2), boss(X_2,X_3), boss(X_4,X_1), father(X_5,X_2), $X_0 \neq X_3$.
Collective Strength	sick(X_0)← father(X_1,X_0), bearded(X_1), boss(X_0,X_2). sick(X_0)← ¬father(X_0,X_1), boss(X_1,X_2), ¬boss(X_2,X_0).

Table 3: Set of rules

	Scoring Function	**Formula**
1	Coverage	$\frac{tp+fp}{N}$
2	Laplace Accuracy	$\frac{tp+1}{tp+fp+2}$
3	Piatetski-Shapiro's	$\frac{tp \cdot tn - fn \cdot fp}{N^2}$
4	ϕ-coefficient	$\frac{tp \cdot tn - fn \cdot fp}{\sqrt{(tp+fn)\cdot(tp+fp)\cdot(fp+tn)\cdot(fn+tn)}}$
5	Support	$\frac{tp}{N}$
6	Rule Accuracy	$\frac{tp}{tp+fp}$
7	Satisfaction	$\frac{tp \cdot tn - fp \cdot fn}{N \cdot (tp+fp) \cdot (tn+fp)}$
8	Confirmation	$\frac{(tp \cdot tn - fp \cdot fn)^2}{N^3 \cdot (tp+fp) \cdot (tn+fp)}$
9	F-measure	$2 \cdot \frac{tp}{2 \cdot tp+fn+fp}$
10	kappa (κ)	$\frac{2 \cdot (tp \cdot tn - fp \cdot fn)}{N^2 - (tp+fn)\cdot(tp+fp) - (fp+tn)\cdot(tn+fn)}$
11	Odds-ratio	$\frac{tp \cdot tn}{fp \cdot fn}$
12	Yule's Q coefficient (Q)	$\frac{tp \cdot tn - fp \cdot fn}{tp \cdot tn + fp \cdot fn}$
13	Lift (Interest)	$\frac{N \cdot tp}{(tp+fp)\cdot(tp+fn)}$
14	Collective Strength	$\frac{tp+tn}{(tp+fp)\cdot(tp+fn)+(fn+tn)\cdot(fp+tn)} \cdot \frac{N^2-(tp+fp)\cdot(tp+fn)-(fn+tn)\cdot(fp+tn)}{N-tp-tn}$
15	Jaccard(ζ)	$\frac{tp}{tp+fp+fn}$

Table 4. List of Scoring functions



sick(b3).	not(sick(a1)).	bearded(a3).	father(a1, b1).	father(b3, c3).	boss(a1, b1).	boss(b3, c4).
sick(b4).	not(sick(a2)).	bearded(a4).	father(a1, b2).	father(b4, c4).	boss(a2, b2).	boss(b4, c4).
sick(c3).	not(sick(a3)).	bearded(c1).	father(a2, b1).	father(b4, c5).	boss(a2, b3).	boss(b4, c5).
sick(c4).	not(sick(a4)).	bearded(c2).	father(a2, b2).	father(c1, d1).	boss(a3, b2).	boss(c1, d1).
sick(c5).	not(sick(b1)).	bearded(c4).	father(a3, b3).	father(c2, d1).	boss(a3, b3).	boss(c2, d1).
sick(d2).	not(sick(b2)).	bearded(d1).	father(a3, b4).	father(c2, d2).	boss(a4, b3).	boss(c3, d2).
sick(d3).	not(sick(c1)).	bearded(d2).	father(a4, b3).	father(c2, d3).	boss(b1, c1).	boss(c3, d3).
sick(d4).	not(sick(c2)).		father(a4, b4).	father(c4, d2).	boss(b2, c2).	boss(c3, d4).
	not(sick(d1)).	smoker(b3).	father(b1, c1).	father(c4, d3).	boss(b2, c3).	boss(c4, d2).
		smoker(b4).	father(b1, c2).	father(c5, d4).	boss(b2, c4).	boss(c4, d3).
		smoker(d2).	father(b2, c3).		boss(b3, c2).	boss(c4, d4).
					boss(b3, c3).	boss(c5, d4).

Table 5. Knowledge Base

References

1. van Rijsbergen C.J.: Information Retrieval. Butterworth, (1979)
2. Brin, S., Motwani, R., Silverstein, C.: Beyond Market Baskets: Generalizing Association Rules to Correlations. SIGMOD Conference. 265-276 (1997)
3. Aggarwal, C.C. Yu, P.S.: A New Framework For Itemset Generation. PODS 98, Symposium on Principles of Database Systems. 18–24 (1998)
4. Mosteller, F.: Association and Estimation in Contingency Tables. Journal of the American Statistical Association. 63, 1–28 (1968)
5. Yule, G.U.: On the Association of Attributes in Statistics. Philosophical Transactions of the R.Society, Ser. A. 194, 257–319 (1900)
6. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement. 20, 37–46 (1960)
7. Flach, P., Lachiche, N.: A First-Order Approach to Unsupervised Learning (1999)
8. Agrawal, R., Imielinski, T., Swami, A.N.: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference. 207–216 (1993)
9. Mooney, R.J., Califf, M.E.: Induction of First-Order Decision Lists: Results on Learning the Past Tense of English Verbs. JAIR. 3, 1–24 (1995)
10. Shapiro, E.Y.: Algorithmix Program Debudding. Cambridge, MA: MIT Press (1983)
11. Srinivasan, A., Muggleton, S.H., King, R.D., Sternberg, M.J.E.: Mutagenesis: ILP experiments in a non-determinate biological domain. Proceedings of the 4th International Workshop on Inductive Logic Programming. 237, 217–232 (1994)
12. Muggleton, S., Feng, C.: Efficient Induction of Logic Programs. Algorithmic Learning Theory. 368-381 (1990)
13. Quinlan, J.R.: Learning First-Order Definitions of Functions. JAIR. 5, 139–161 (1996)
14. Fürnkranz, J.: FOSSIL: A Robust Relational Learner. European Conference on Machine Learning. 122–137 (1994)
15. Ali, K.M., Pazzani, M.J.: HYDRA: A Noise-tolerant Relational Concept Learning Algorithm. IJCAI. 1064–1071 (1993)
16. Lavrac, N., Dzeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood. (1994)
17. Wogulis, J., Pazzani, M.J.: A Methodology for Evaluating Theory Revision Systems: Results with Audrey II. IJCAI. 1128–1134 (1993)
18. Lavrac. N, Flach, P.A., Zupan, B.: Rule Evaluation Measures: A Unifying View. International Workshop on Inductive Logic Programming. 174–185 (1999)
19. Quinlan, J.R.: FOIL: A Midterm Report. European Conference on Machine Learning. 3–20 (1993)
20. Agresti, A.: Categorical Data Analysis. Wiley-Interscience. (1990)
21. Piatetsky-Shapiro, G.: Discovery, Analysis, and Presentation of Strong Rules. 229–248 (1991)
22. Clark, P., Boswell, R.: Rule Induction with CN2: Some Recent Improvements. EWSL, 151–163 (1991)
23. Tan, P., Kumar, V., Srivastava, J.: Selecting the right objective measure for association analysis. Inf. Syst. 4, 293–313 (2004)
24. Pazzani, M.J., Brunk, C., Silverstein, G.: A Knowledge-intensive Approach to Learning Relational Concepts. ICML. 432–436 (1991)
25. Rissanen J.: Modeling by Shortest Data Description. Automatica, 14, (1978)

26. GÓMEZ, A.J. y FERNANDEZ, G. : Inducción de definiciones lógicas a partir de relaciones: mejoras en los heurísticos del sistema FOIL. PRODE, 292–302 (1992)
27. Freitag, D. : Information Extraction from HTML: Application of a General Machine Learning Approach. AAAI, 517–523 (1998)