Machine Learning manuscript No.
(will be inserted by the editor)

Relational Information Gain

Marco Lippi · Manfred Jaeger · Paolo Frasconi · Andrea Passerini

Received: date / Accepted: date

Abstract We introduce relational information gain, a refinement scoring function measuring the informativeness of newly introduced variables. The gain can be interpreted as a conditional entropy in a well-defined sense and can be efficiently approximately computed. In conjunction with simple greedy general-to-specific search algorithms such as FOIL, it yields an efficient and competitive algorithm in terms of predictive accuracy and compactness of the learned theory. In conjunction with the decision tree learner TILDE, it offers a beneficial alternative to lookahead, achieving similar performance while significantly reducing the number of evaluated literals.

Keywords Relational Learning · Inductive Logic Programming · Information Gain

1 Introduction

Many ILP or relational learning systems build discriminative models by a stepwise refinement of logical-relational features. For example, in general-to-specific rule learners like FOIL [16], features are the bodies of Horn clauses that are constructed by adding one literal at a time. In models that adopt a decision-tree style design (in a wide sense), like TILDE [3], Multi-Relational Decision Trees [11], Relational Probability Trees [13], or Type Extension Trees (TETs) [8], features are represented by branches in the tree structure, which are constructed in an iterative top-down process.

A distinguishing characteristic of incremental feature construction in relational learning is the possibility to refine a current feature for a given set of entities X by introducing new entities Y and their attributes via relations $r(X, Y)$.

Marco Lippi
Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, Italy
E-mail: lippi@dsi.unifi.it

M. Jaeger
Department for Computer Science, Aalborg University, Denmark

P. Frasconi
Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, Italy

A. Passerini
Dipartimento di Ingegneria e Scienza dell'Informazione, Università degli Studi di Trento, Italy

The search for the best feature refinement is typically directed by some scoring function that evaluates its usefulness for discriminating the class label of X. A refinement that does not introduce any new entities can be scored in a relatively straightforward manner using standard information gain metrics. A refinement introducing new entities is more difficult to evaluate, however: standard metrics can measure the *direct informativeness* of such a refinement, i.e., the direct improvement in the feature's discriminative power. However, it is widely recognized that the main benefit of introducing Y is not always its direct informativeness, but the possibility it opens up to construct, in further refinement steps, informative features for X by imposing suitable conditions on Y. Two main approaches have been used to take into account this *potential informativeness* of a literal introducing new entities. A first approach evaluates *determinate literals* [15], which are literals where for each X there exists exactly one Y with $r(X, Y)$. Determinate literals are not directly informative, but their inclusion in the clause is computationally relatively inexpensive, which is why, e.g., FOIL adds all possible determinate literals to a clause in order to exploit their potential informativeness. A second approach consists of *lookahead* techniques [2, 4, 18], where for the scoring of the literal $r(X, Y)$ already further possible refinement steps using Y are considered. Both determinate literals and lookahead have severe limitations: the former represents only a very special kind of potentially informative literals, and the latter is subject to a combinatorial search space explosion when performing lookahead over multiple refinement steps (which is why, in practice, lookahead may need to be constrained to certain user-defined refinement patterns).

The goal of this paper is to develop a notion of *relational information gain (RIG)* for scoring candidate literals that introduce new variables, such that both direct and potential informativeness can be measured. Specifically, we have the following desiderata for RIG:

1. RIG captures a sound and general information theoretic concept of reduction in conditional entropy of the class label distribution. It thereby is widely applicable, and not a specialized heuristic scoring function for a specific model or search strategy.
2. RIG increases as a function of the *direct informativeness* of a literal $r(X, Y)$, defined as the information about the target relation associated with the existence of X, Y such that $r(X, Y)$ (or, more generally, with the number of such pairs).
3. RIG increases as a function of the *potential informativeness* of a literal $r(X, Y)$, defined as the maximum information that can be gained about the target relation thanks to the introduction of Y via $r(X, Y)$ and further refinements using Y (without lookahead, only based on the immediate relational properties of r).

In the following sections we develop a RIG score that is motivated by these desiderata.

2 Data: relational and pseudo-iid

Since information gain is a statistical concept based on a probabilistic data model, we first investigate what kind of statistical model of relational data is appropriate to support the definition of RIG. We assume that the data consists of a single relational or logical database containing constants $c_1, \ldots, c_n$, attributes $a_1, \ldots, a_k$, and relations $r_1, \ldots, r_l$. We use $\mathcal{C}$ to denote the set of constants. When taking a more semantic view, we may also refer to the elements of $\mathcal{C}$ as *domain elements* or *entities*. The data set can be identified with an interpretation function I that assigns to each ground atom $a_i(c)$ a value in the range of attribute a_i, and to each ground atom $r_j(\boldsymbol{c})$ a truth value $true, false$ (throughout we use bold font to denote tuples of constants or variables). We can therefore write a data set as a pair

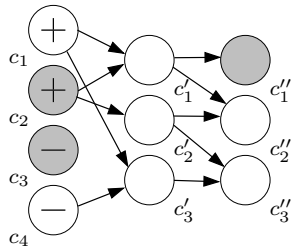

Fig. 1 A small relational structure

$\mathcal{D} = (\mathcal{C}, I)$, and view it as a Herbrand interpretation (in the slightly generalized sense that we allow non-boolean attributes).

Based on this logical view of relational data, we can use for a logical expression ϕ in the relations of $\mathcal{D}$ the notation $\mathcal{D} \models \phi$ to say that ϕ is true in $\mathcal{D}$.

Usually, there will be a distinguished *class* attribute or relation. For notational simplicity we will assume that *class* is binary with values *positive* (+) and *negative* (−). The domain $\mathcal{C}$ will usually be partitioned into sets of constants representing different types of objects, and the arguments of attributes and relations will also be typed. However, for notational simplicity we will not make such type constraints explicit.

Figure 1 shows a graphical representation of a relational data set with 10 constants, a class attribute, one further Boolean attribute a (elements for which $a = true$ are represented by grey shading), and one binary relation r (indicated by the edges in the graph). The *class* attribute here only applies to entities of a type that just comprises $c_1, \ldots, c_4$.

We note that in this data model we can also accommodate the case where the data consists of several relational structures, each representing one observation or example (the "learning from interpretations" setting [5], which is appropriate, e.g., for molecular data, where each example corresponds to a distinct molecule). Such data can be encoded as a single structure by introducing constants representing the examples, and adding to each relation an argument for the example identifier: if, e.g., *contains(sulfur)* is true in the 11th example (molecule), then this becomes *contains(*e_{11}*,sulfur)*.

In a probabilistic interpretation of the data, one will view the ground atoms of the Herbrand base as random variables, and the observed Herbrand interpretation as a draw from the joint distribution of these atoms. It is a distinctive feature of relational learning that one does not want to make strong assumptions of independence and identical distributions for these random variables. This leads to some limitations for the applicability of standard statistical and information-theoretic methods. For example, to apply notions of information gain, one would first have to be able to estimate the entropy of the class label distribution. However, without independence assumptions for the random variables *class*(c_1), *class*$(c_2), \ldots,$ *class*(c_n) one cannot use the observed empirical frequencies of *pos* and *neg* labels to estimate properties of the class label distribution, including its entropy.

It seems that in order to leverage certain types of statistical analysis tools, one actually has to compromise the holistic relational data model, and extract from the overall relational structure a number of separate sub-structures, which are then treated as iid samples. Such a transformation of relational data into a collection of *pseudo-iid* data fragments is performed in various ways by several relational learning systems. For example the *local training sets* employed by FOIL can be seen in this way; the learning routines in the *Proximity* system (`http://kdl.cs.umass.edu/software`) operate on collections of sub-graphs extracted from the underlying database. Most notably, perhaps, *proposition-*

alization approaches to relational learning can be seen as consisting of a construction of a pseudo-iid data view in the sense of the following definition, and the subsequent application of a standard learner for attribute-value data on this data view (see [6, Chapter 4] for an overview of propositionalization).

Before formally defining pseudo-iid data views, we introduce some notational preliminaries. Uppercase letters $X, X_1, X_2, Y, \ldots$ are used to denote variables. Tuples of variables are denoted in boldface $\boldsymbol{X}, \boldsymbol{Y}, \ldots$. Tuples are also seen as the set of variables they contain, and set-theoretic notation like $\boldsymbol{X} \cup \boldsymbol{Y}, \boldsymbol{X} \setminus \boldsymbol{Y}, \ldots$ can be used to construct new sets of variables. A substitution of constants for variables $\boldsymbol{X}$ is a mapping $\boldsymbol{X} \to \mathcal{C}$, and $\mathcal{C}^{\boldsymbol{X}}$ is the set of all such substitutions.

We use $\theta, \theta', \ldots$ to denote substitution mappings. The concrete substitution that maps $\boldsymbol{X}$ to the tuple of constants $\boldsymbol{c}$ is denoted $\boldsymbol{X}/\boldsymbol{c}$. We write $F(\boldsymbol{Y})[\theta]$ for the result of performing substitution θ on the free variables $\boldsymbol{Y}$ of a formula F. When θ, θ' are substitutions that are defined on disjoint domains of variables, we can write $F(\boldsymbol{Y})[\theta, \theta']$ for the result of performing both substitutions.

Definition 1 Let $\mathcal{D} = (\mathcal{C}, I)$ be a relational data set. A *pseudo-iid (p-iid) data view* of $\mathcal{D}$ consists of

i an integer m, and variables $\boldsymbol{X} = X_1, \ldots, X_m$.
ii a set of examples, where each example consists of a substitution $\theta \in \mathcal{C}^{\boldsymbol{X}}$. We usually write examples as tuples $\boldsymbol{c}_i = (c_{i_1}, \ldots, c_{i_m})$, where it is understood that c_{i_j} is the substitution value for X_j.
iii A set of attributes $F(X_1, \ldots, X_m)$ defined on $\mathcal{C}^{\boldsymbol{X}}$, where an attribute can be any function (e.g., boolean, integer-, or real-valued).

We write *dv*$(X_1, \ldots, X_m)$ to denote a p-iid data view in the variables $X_1, \ldots, X_m$.

Part (iii) of this definition is extremely general. Usually, one will only consider attributes that only depend on the relational structure of $\mathcal{D}$, and, thus, are invariant under renaming of the constants. However, for the purpose of the present paper we need not formalize these natural restrictions.

Table 1 shows two possible extractions of p-iid data views from the relational structure of Figure 1. In the first view, $m = 1$. The first two attributes in this data view are just the original attributes *class*$(X_1), a(X_1)$ given in the data. The third attribute, denoted $\exists Y r(X_1, Y)$, is a boolean attribute that has value $true$ for an example c_i if $\mathcal{D} \models \exists Y r(c_i, Y)$. The attribute $\#Y r(X_1, Y)$ is integer-valued, and it represents, for example c_i , the number of entities c' for which $r(c_i, c')$ is true. Table 1 (b) shows a p-iid data view of the same relational data with $m = 2$ obtained by selecting all r-connected pairs of entities with the first component being one of $c_1, \ldots, c_4$. The first four columns here are attributes that are directly obtained from the attributes and relations in the data. The last two attributes are derived attributes, similar to the ones in Table 1 (a). Note that attributes in a pseudo-iid data view can be *internal* attributes of the example tuples, i.e., attributes whose values is determined only by the substructure induced by the example m-tuple (the first two, respectively four attributes in Table 1 (a) and (b)), or *embedding* attributes whose values depend on the relational connections between the example tuples and the rest of the domain (the last two attributes in both tables).

Data-tables such as the ones shown in Table 1 could be generated from the underlying relational data for any number of purposes. We refer to them as p-iid data views when the tables are used to support operations that usually require an iid assumption. For the purpose of this paper, this is mostly the computation of entropies and conditional entropies between

Table 1 Pseudo-iid relational data

(a)

Example X_1	*class*(X_1)	$a(X_1)$	$\exists Yr(X_1, Y)$	$\#Yr(X_1, Y)$
c_1	+	f	t	2
c_2	+	t	t	2
c_3	−	t	f	0
c_4	−	f	t	1

(b)

Example X_1, X_2	*class*(X_1)	$a(X_1)$	$r(X_1, X_2)$	$r(X_2, X_1)$	$\exists Yr(X_2, Y)$	$\#Yr(X_2, Y)$
c_1, c_1'	+	f	t	f	t	2
c_1, c_3'	+	f	t	f	t	1
c_2, c_1'	+	t	t	f	t	2
c_2, c_2'	+	t	t	f	t	2
c_4, c_3'	−	f	t	f	t	1

different data columns. In other cases, a p-iid data view may underly the split of a relational data set into test and training sets. The implicit iid assumption for p-iid data views can be technically incorrect (e.g., in Table 1 (b) the attribute $a(X_1)$ corresponds to the same random variable $a(c_1)$ in the first and second example, and so, with probability one, this attribute has the same value in the two examples), but may also actually hold (e.g., when the examples in the p-iid data view correspond to the original iid observations of distinct relational structures).

As mentioned above, an important example of p-iid data views are tables created in propositionalization approaches to relational learning. It is important to note, however, that when we talk about p-iid data views we are not implicitly assuming a propositionalization approach to relational learning. P-iid data views are only a conceptual model that provide the foundation for the application of certain statistical operations. It is not assumed that any p-iid data views are explicitly generated and operated on by the learner (even though the local training sets of FOIL, and the local example sets in TILDE can be seen as 'materialized' p-iid data views). Furthermore, it is not assumed that the attributes in the p-iid data views we consider are the attributes actually available for the final model. For example, the definition of relational information gain that we propose is based on p-iid data views containing attributes of the form defined by equations (6) and (7) below. However, these views are only used to justify the definition of a particular refinement scoring function for use in existing rule learning systems. In the actual learning process, no tables containing values for these attributes are constructed, and the learned model cannot use the attributes (6) and (7) directly.

3 Relational Information Gain

Our goal is to set up a general framework for measuring information gain of refinements. The p-iid data views introduced in the previous section serve as the basis for information theoretic concepts. We now proceed to relate p-iid data views with refinement steps in inductive learners, and to use information gain measures computed on p-iid data views to score candidate refinements.

We assume that the following is given in a learning scenario for scoring a refinement:

– A p-iid data view *dv*$(X_1, \ldots, X_m)$.

– A current query $Q(\boldsymbol{Y})$, i.e., a conjunction of literals jointly containing variables $\boldsymbol{Y}$.

We refer to a pair $dv(\boldsymbol{X}), Q(\boldsymbol{Y})$ as a *refinement scenario*. A *candidate refinement* in a refinement scenario is

– A literal $l(\boldsymbol{Z})$ containing variables $\boldsymbol{Z}$. We call $l(\boldsymbol{Z})$ a *basic refinement* of $Q(\boldsymbol{Y})$ if $\boldsymbol{Z} \subseteq \boldsymbol{X} \cup \boldsymbol{Y}$; otherwise $l(\boldsymbol{Z})$ is called a *variable introduction refinement (VI-refinement)*. We write $\boldsymbol{Z}_{new} := \boldsymbol{Z} \setminus (\boldsymbol{X} \cup \boldsymbol{Y})$ for the new variables introduced by $l(\boldsymbol{Z})$.

Example 1 Suppose we use FOIL to learn a model for classifying the entities in the relational structure of Figure 1. The initial learning situation is given by the labeled entities, i.e., the p-iid data view $dv(X_1)$ given by the first two columns of Table 1 (a), and an empty query $Q(\boldsymbol{Y}) = \emptyset$ (i.e., body of the clause under construction). The literal $l(X_1) = a(X_1)$ then is a basic refinement, whereas $l(X_1, X_2) = r(X_1, X_2)$ is a VI-refinement. When refining with $r(X_1, X_2)$ (i.e., constructing the clause $class(X_1) \leftarrow r(X_1, X_2)$), FOIL constructs a new local training set, which is a new p-iid data view $dv(X_1, X_2)$ consisting of the first two columns of Table 1 (b). The refinement scenario in the next step then consists of this data view and the query $Q(X_1, X_2) = r(X_1, X_2)$.

Now consider using TILDE to learn a logical decision tree. The initial learning situation is analogous as in FOIL, with Table 1 (a) specifying the initial set of examples. After refining (the empty query) with $r(X_1, X_2)$ a new node is constructed with the associated set of examples

Example X_1	*class*
c_1	$+$
c_2	$+$
c_4	$-$

(1)

which is the p-iid data view $dv'(X_1)$ in the next refinement scenario (also consisting of the query $Q(X_1, X_2) = r(X_1, X_2)$).

Even though the p-iid data views are different when FOIL or TILDE go through the same sequence of refinement steps, the definitions coincide with regard to which refinements are basic or VI-refinements.

A candidate refinement $l(\boldsymbol{Z})$ gives rise to a Boolean attribute in the p-iid data view that represents whether $\exists \boldsymbol{Y}\boldsymbol{Z} Q(\boldsymbol{Y}), l(\boldsymbol{Z})$ is true for example $\boldsymbol{c} \in dv(\boldsymbol{X})$. Since $\boldsymbol{Y}$ and $\boldsymbol{Z}$ may also contain some of the variables from $\boldsymbol{X}$ for which we substitute the example $\boldsymbol{c}$, we need to write this feature more precisely as $\exists(\boldsymbol{Y} \cup \boldsymbol{Z}) \setminus \boldsymbol{X}\ Q(\boldsymbol{Y}), l(\boldsymbol{Z})$. Thus, we define (relative to a given refinement context $dv(\boldsymbol{X}), Q(\boldsymbol{Y})$):

$$F^{exists}_{l(\boldsymbol{Z})}(\boldsymbol{c}) := \begin{cases} true & \text{if } \exists \theta \in \mathcal{C}^{\boldsymbol{Z}_{new}} : \mathcal{D} \models \exists \boldsymbol{Y} \setminus \boldsymbol{X} :\ Q(\boldsymbol{Y})[\theta, \boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\theta, \boldsymbol{X}/\boldsymbol{c}] \\ false & \text{otherwise} \end{cases} \quad (2)$$

The condition in the *true* case of the above definition can be equivalently expressed as $\mathcal{D} \models \exists(\boldsymbol{Y} \cup \boldsymbol{Z}) \setminus \boldsymbol{X} :\ Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\boldsymbol{X}/\boldsymbol{c}]$. The asymmetric treatment of the existential quantification over the variables $\boldsymbol{Z}_{new}$ and $\boldsymbol{Y} \setminus \boldsymbol{X}$ in (2) is motivated by the fact that in this way we obtain a definition that is more uniform with the following one. This second type of feature that we now introduce is more informative than F^{exists}. It returns the actual number of substitutions for $\boldsymbol{Z}_{new}$ that make $Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\boldsymbol{X}/\boldsymbol{c}]$ true:

$$F^{count}_{l(\boldsymbol{Z})}(\boldsymbol{c}) := \left| \{\theta \in \mathcal{C}^{\boldsymbol{Z}_{new}} \mid \mathcal{D} \models \exists \boldsymbol{Y} \setminus \boldsymbol{X} :\ Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\theta, \boldsymbol{X}/\boldsymbol{c}] \} \right| \quad (3)$$

Based on F^{count} or F^{exists} features one can score candidate refinements $l(\boldsymbol{Z})$ by standard information gain

$$ig(F_{l(\boldsymbol{Z})}) = H(class) - H(class \mid F_{l(\boldsymbol{Z})}) \tag{4}$$

where $H(\cdot)$ and $H(\cdot|\cdot)$ denote the entropy function and the conditional entropy function, respectively. We omit the superscripts *count* or *exists* when statements or definitions apply uniformly to both versions. The formal definitions of the F^{count} and F^{exists} features apply to basic and VI-refinements. For basic refinements F^{count} degenerates to a 0/1-valued equivalent of $F^{exists}_{l(\boldsymbol{Z})}$. Scoring refinements based on F^{count} only is appropriate when the models constructed from the resulting query $Q(\boldsymbol{Y}), l(\boldsymbol{Z})$ can actually use the quantitative information of counts of substitutions, e.g., TILDE with counting literals [19], Relational Probability Trees [13], Markov Logic Networks [17], and Type Extension Trees [8]. Classic ILP systems with existential semantics for newly introduced variables, on the other hand, should score all candidate refinements using F^{exists}.

So far, we have set up a framework in which refinements can be scored based on standard information theoretic principles. $ig(F_{l(\boldsymbol{Z})})$ scores the direct informativeness of the refinement $l(\boldsymbol{Z})$, i.e., the improvement of discriminative power for the class label from the new literal alone, without any further refinements. Moreover, $ig(F^{count})$ can measure the direct informativeness presented by *degree disparity*, i.e., the situation in which positive examples tend to have more (or fewer) relational neighbors connected via $l(\boldsymbol{Z})$ than negative examples.

We now extend this approach to also measure potential informativeness of VI-refinements. To motivate our approach, consider the case where the current refinement scenario consists of a data view $dv(X)$, $Q(\boldsymbol{Y}) = \emptyset$ (as usually the case in the first step of the induction), and we consider the VI-refinement $r(X, Z)$. Suppose that $r(X, Z)$ is a *determinate literal*, i.e., for each $c \in dv(X)$ there exists exactly one $c' \in \mathcal{C}$ with $\mathcal{D} \models r(c, c')$. We can now consider the set of domain entities that are associated via r with positive and negative examples, respectively:

$$\begin{aligned} B^+ &:= \{c' \in \mathcal{C} \mid \exists c \in dv(X) \text{ with } class(c) = + \text{ and } r(c, c')\} \\ B^- &:= \{c' \in \mathcal{C} \mid \exists c \in dv(X) \text{ with } class(c) = - \text{ and } r(c, c')\} \end{aligned} \tag{5}$$

If the two sets B^+, B^- are sufficiently distinct, then we may be able to discriminate between positive and negative examples with a refined query $r(X, Z), a(Z)$, where $a(Z)$ is an attribute on Z that is correlated with membership in B^+ (or B^-).

Figure 2 shows three different data sets, where in all cases the labeled nodes $c_1, \ldots, c_6$ represent the entities in a current p-iid data view, and the arrows represent a binary relation r. In (a) and (b) r is determinate. In (a) the sets B^+ and B^- are disjoint, and if an attribute (or conjunction of attributes) can be found that is true for c'_1 but not for c'_2, then we would be able to perfectly classify the examples. In (b), in contrast $B^+ = B^-$, and, moreover, positive and negative examples have exactly the same properties with regard to r, so that the refinement with $r(X, Z)$ is neither directly nor potentially informative. In Figure 2 (c) r is not determinate. Nevertheless, the sets $B^+ = \{c'_1, c'_2\}$ and $B^- = \{c'_2, c'_3\}$ are still defined by (5). While the two sets are not disjoint, an attribute $a(Z)$ characterizing membership in B^+ would still allow us to split with the query $r(X, Z), a(Z)$ the data set into $\{c_1, c_2, c_3, c_4\}$ and $\{c_5, c_6\}$. However, characterizing membership in B^+ here would not be optimal: if $a(Z)$ characterized membership in $B = \{c'_1\}$ instead, then $r(X, Z), a(Z)$ would yield a perfect split.

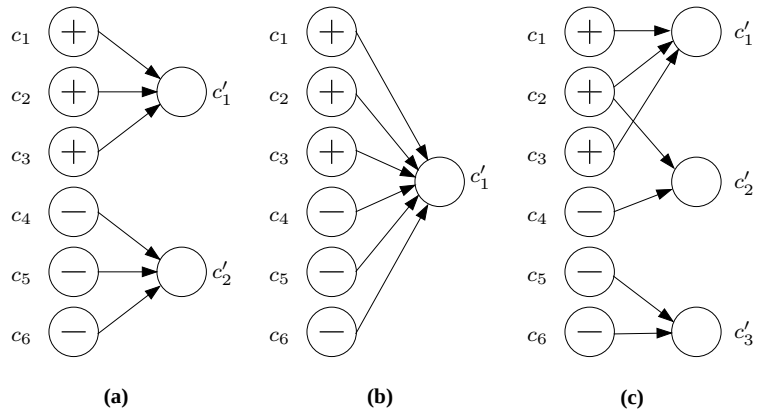

Fig. 2 Potential informativeness via determinate (a),(b) and general (c) relations

Table 2 Values for selected F^{exists} attributes

Example X	$F^{exists}_{r(X,Z),B^+}$ (a)	$F^{exists}_{r(X,Z),B^+}$ (b)	$F^{exists}_{r(X,Z),B^+}$ (c)	$F^{exists}_{r(X,Z),\{c'_1\}}$ (c)
c_1	t	t	t	t
c_2	t	t	t	t
c_3	t	t	t	t
c_4	f	t	t	f
c_5	f	t	f	f
c_6	f	t	f	f

In general, for any set $B \subseteq \mathcal{C}$ we can define the Boolean feature $F^{exists}_{r(X,Z),B}(c)$ that represents whether c has any r-successor in B. Table 2 gives the resulting values for $B = B^+$ in the three data sets of Figure 2. For data set (c) also the value for $B = \{c'_1\}$ is shown.

The main idea on which we build our definition of relational information gain, now, is that the refinement $r(X, Z)$ is potentially informative if there exists at least some set B for which $F^{exists}_{r(X,Z),B}$ is informative. To cast this into a formal definition, we first define the $F^{exists}_{l(\boldsymbol{Z}),B}$ attribute in the full generality for arbitrary refinement scenarios. In the general case where the VI-refinement $l(\boldsymbol{Z})$ introduces more than one new variable, we have to consider subsets B of possible substitutions for all new variables, i.e., $B \subseteq \mathcal{C}^{\boldsymbol{Z}_{new}}$, rather than just $B \subseteq \mathcal{C}$. For any such B we define as a small modification of (2):

$$F^{exists}_{l(\boldsymbol{Z}),B}(\boldsymbol{c}) := \begin{cases} true & \text{if } \exists \theta \in B : \ \mathcal{D} \models \exists \boldsymbol{Y} \setminus \boldsymbol{X} : \ Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\theta, \boldsymbol{X}/\boldsymbol{c}] \\ false & \text{otherwise} \end{cases} \tag{6}$$

As before, this definition is relative to a given refinement scenario, and we also define the more informative count version:

$$F^{count}_{l(\boldsymbol{Z}),B}(\boldsymbol{c}) := |\{\theta \in B \mid \mathcal{D} \models \exists \boldsymbol{Y} \setminus \boldsymbol{X} : \ Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\theta, \boldsymbol{X}/\boldsymbol{c}]\}| \tag{7}$$

For $B = \mathcal{C}^{\boldsymbol{Z}_{new}}$ (6) and (7) become our measures (2), respectively (3) for direct informativeness. Finally, we can now define relational information gain as a measure for direct

Table 3 Feature values in Example 3

(a)

X_1	*class*	$F^{count}_{r(X_1,X_2)}$	$F^{exists}_{r(X_1,X_2)}$	$F^{exists}_{r(X_1,X_2),\{c'_1,c'_2\}}$
c_1	+	2	t	t
c_2	+	2	t	t
c_3	−	0	f	f
c_4	−	1	t	f

(b)

X_1	*class*	$F^{count}_{r(X_2,X_3)}$	$F^{exists}_{r(X_2,X_3)}$	$F^{exists}_{r(X_2,X_3),\{c''_1\}}$
c_1	+	3	t	t
c_2	+	3	t	t
c_4	−	1	t	f

and potential informativeness by maximizing over the information gains obtained from the features $F_{l(\boldsymbol{Z}),B}$ for any possible set B:

Definition 2 Let $dv(\boldsymbol{X}), Q(\boldsymbol{Y})$ be a refinement scenario. For a candidate refinement $l(\boldsymbol{Z})$ we define the existential and count version of *relational information gain* as follows:

$$\mathit{RIG}^{\mathit{exists}}(l(\boldsymbol{Z})) := \max_{B \subseteq \mathcal{C}^{\boldsymbol{Z}_{new}}} \mathit{ig}(F^{\mathit{exists}}_{l(\boldsymbol{Z}),B}) \tag{8}$$

$$\mathit{RIG}^{\mathit{count}}(l(\boldsymbol{Z})) := \max_{B \subseteq \mathcal{C}^{\boldsymbol{Z}_{new}}} \mathit{ig}(F^{\mathit{count}}_{l(\boldsymbol{Z}),B}). \tag{9}$$

Thus, in the definition of *RIG* we are taking an optimistic attitude towards evaluating potential informativeness (or rather, stressing the "potential") by basing the definition on the most discriminating subset B, even though we do not know whether we will be able to characterize this optimal B by further refinements using the available attributes and relations.

Example 2 Consider the candidate refinement $r(X, Z)$ for the refinement scenarios given by the data views represented by Figure 2 (i.e., taking $c_1, \ldots, c_6$ as labeled p-iid examples), and $Q(\boldsymbol{Y}) = \emptyset$. For Figure 2 (a) we have that for $B = \{c'_1\}$ $H(\mathit{class} \mid F^{exists}_{r(X,Z),B}) = H(\mathit{class} \mid F^{count}_{r(X,Z),B}) = 0$, so both $\mathit{RIG}^{\mathit{exists}}(r(X, Z))$, and $\mathit{RIG}^{\mathit{count}}(r(X, Z))$ obtain the maximal possible value $H(\mathit{class})$.

For Figure 2 (b) we have that for all B both $F^{exists}_{r(X,Z),B}$ and $F^{count}_{r(X,Z),B}$ are constant for all c_i (being $true/false$, respectively 0/1, depending on whether B contains c'_1), so all $F_{r(X,Z),B}$ have zero information gain, and hence $\mathit{RIG}^{\mathit{exists}}(r(X, Z)) = \mathit{RIG}^{\mathit{count}}(r(X, Z)) = 0$.

In Figure 2 (c) the maximal $\mathit{ig}(F_{r(X,Z),B})$ scores are attained for $B = \{c'_1\}$. As in (a), this leads to the maximal possible $\mathit{RIG}^{\mathit{exists}}$ and $\mathit{RIG}^{\mathit{count}}$ scores.

Example 3 We again consider the p-iid data view of the relational data in Figure 1, as represented by the first two columns of Table 1 (a), together with the initial query $Q(\boldsymbol{Y}) = \emptyset$. We consider the candidate refinement $r(X_1, X_2)$. The third column in Table 3 (a) shows the feature values for $F^{\mathit{count}}_{r(X_1,X_2)}$, i.e., our measure for the direct informativeness of this refinement when quantitative information can be used. Since the values for the positive examples are distinct from the values for the negative examples, we have that $\mathit{ig}(F^{\mathit{count}}_{r(X_1,X_2)}) = H(\mathit{class})$ is the maximal possible, and hence also $\mathit{RIG}^{\mathit{count}}(r(X_1, X_2))$ is the maximal possible. The feature $F^{\mathit{exists}}_{r(X_1,X_2)}$ provides some information gain, but not a perfect split of the examples.

Thus, for systems with an existential semantics, the refinement $r(X_1, X_2)$ has some direct information value, but does not yet lead to a perfect classification. However, the feature $F^{exists}_{r(X_1,X_2),\{c'_1,c'_2\}}$ has maximal information gain, which leads to a maximal value of RIG^{exists}. This shows that beyond its direct informativeness there also is some additional potential informativeness in the refinement $r(X_1, X_2)$.

Suppose, now, that the initial query is refined by $r(X_1, X_2)$, and that the next refinement scenario is given by the $dv(X_1)$ given by the first two columns of Table 3 (b), and the query $r(X_1, X_2)$ (this corresponds to a sub-problem that would be generated by TILDE). We now consider the candidate refinement $r(X_2, X_3)$. The feature $F^{count}_{r(X_2,X_3)}(c_i)$, in this scenario, represents the number of distinct entities reachable from c_i via a path of length 2. This feature, again, has maximal information gain, and so a maximal possible RIG^{count} value is obtained (however, if we were using a system that can utilize quantitative count features, then the induction would probably have ended after the first refinement, and this score is not very relevant any more). The feature $F^{exists}_{r(X_2,X_3)}(c_i)$, on the other hand, has zero information gain, showing that for systems with existential semantics the refinement $r(X_2, X_3)$ is not directly informative. However, $F^{exists}_{r(X_2,X_3),\{c''_1\}}(c_i)$, again, has maximal information gain, indicating its potential informativeness. If the refinement is chosen, then in the next step it would be found that the basic refinement $a(X_3)$ has maximal information gain, and that the query $r(X_1, X_2), r(X_2, X_3), a(X_3)$ provides a perfect classification rule.

3.1 RIG in Practice

In this section we discuss several important properties of RIG, especially those that have a direct impact on an effective implementation of RIG score in existing relational learners.

3.1.1 Direct vs. potential informativeness

The information gain from the non-B-conditioned features (2) and (3) can be seen as measures for direct informativeness (for systems using purely logical and quantitative semantics, respectively). Since these values are included in the maximizations (8) and (9) via $B = \mathcal{C}^{\boldsymbol{Z}_{new}}$, we obtain that $RIG(l(\boldsymbol{Z}))$ scores are lower bounded by the direct informativeness scores $ig(F_{l(\boldsymbol{Z})})$. Furthermore, the ratio $ig(F_{l(\boldsymbol{Z})})/RIG(l(\boldsymbol{Z})) \in [0, 1]$ shows to what extent the RIG score is based on potential informativeness. When the ratio is 0, then the RIG score is entirely due to potential informativeness. This can be used in various ways to guide the refinement search.

First, one can penalize or reward candidate refinements with a low ig/RIG ratio, thus allowing a choice between exploitation (prefer refinements with direct information value) and exploration (encourage refinements that lead to an extended search along chains of relations).

Second, a low ig/RIG ratio means that $l(\boldsymbol{Z})$ only is useful when further conditions on $\boldsymbol{Z}_{new}$ are imposed. This should influence the way new candidate refinements are constructed. Specifically, it can be useful then to force the next refinement to explore the potential informativeness of $\boldsymbol{Z}_{new}$, i.e., to allow only literals that contain $\boldsymbol{Z}_{new}$. Without such a search bias, it can happen that in a purely greedy construction successive refinements $l(\boldsymbol{Z}), l(\boldsymbol{Z}'), l(\boldsymbol{Z}''), etc.$ are chosen which are merely syntactic variants of the same literal (differing only in the names of the new variables). Intuitively, multiple (equivalent) lines of possible exploration are opened, without pursuing any of them. This phenomenon does

Table 4 Approximate *RIG* computation

1. **init:** $B = \emptyset$
2. *Pos_constants*=$\emptyset$; *Neg_constants*=$\emptyset$
3. *Pos_score*=0; *Neg_score*=0
4. *Candidate_constants* = $\{\boldsymbol{c}_{new} \in \mathcal{C}^{\boldsymbol{Z}_{new}} \mid \exists \boldsymbol{c} \in dv(\boldsymbol{X}) : \mathcal{D} \models \exists \boldsymbol{Y} \setminus \boldsymbol{X}\, Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\boldsymbol{X}/\boldsymbol{c}, \boldsymbol{Z}_{new}/\boldsymbol{c}_{new}]\}$
5. **for all** $\boldsymbol{c}_{new} \in$ *Candidate_constants*
6. $score(\boldsymbol{c}_{new}) = ig(F_{l(\boldsymbol{Z}),\{\boldsymbol{c}_{new}\}})$
7. **if** $P(class(\boldsymbol{X}) = + \mid F_{l(\boldsymbol{Z}),\{\boldsymbol{c}_{new}\}} = true) > P(class(\boldsymbol{X}) = +)$
8. $Pos\_constants = Pos\_constants \cup \{\boldsymbol{c}_{new}\}$
9. $Pos\_score = Pos\_score + score(\boldsymbol{c}_{new})$
10. **else**
11. $Neg\_constants = Neg\_constants \cup \{\boldsymbol{c}_{new}\}$
12. $Neg\_score = Neg\_score + score(\boldsymbol{c}_{new})$
13. **if** $Pos\_score > Neg\_score$
14. **for** $\boldsymbol{c}_{new} \in$ *Pos_constants* in decreasing order of $score(\boldsymbol{c}_{new})$
15. **if** $ig(F_{l(\boldsymbol{Z}),B\cup\{\boldsymbol{c}_{new}\}}) > ig(F_{l(\boldsymbol{Z}),B})$ $B := B \cup \{\boldsymbol{c}_{new}\}$
16. **else** ... // same as 14.,15 with *Neg* for *Pos*
17. **return** $ig(F_{r,B})$ // as approximation of $RIG(r)$

not indicate a fundamental problem with RIG scoring. It only shows that scoring functions that take into account potential informativeness may need to be combined with somewhat different search strategies than other scoring functions.

3.1.2 Computing RIG

The definition of RIG includes a maximization over all subsets of $\mathcal{C}^{\boldsymbol{Z}_{new}}$, which is presumably computationally intractable (the exact complexity of computing RIG is an open problem). In our implementation we use an approximate method for computing RIG scores. The algorithm shown in Table 3.1.2 constructs a set B that approximates the *argmax* in (8), respectively (9).

First we observe that we only need to consider for inclusion in B tuples $\boldsymbol{c}_{new} \in \mathcal{C}^{\boldsymbol{Z}_{new}}$ that make the extended query $Q(\boldsymbol{Y}), l(\boldsymbol{Z})$ true for at least one example $\boldsymbol{c} \in dv(\boldsymbol{X})$ (line 4.). Other tuples can have no impact on the feature values $F_{l(\boldsymbol{Z}),B}$, and thus are irrelevant for the RIG scores.

The construction consists of two main steps: in the first step we determine whether the set B should be composed of tuples that are mostly associated with positive examples, or of tuples that are mostly associated with negative examples, i.e., whether we attempt to obtain a feature $F_{l(\boldsymbol{Z}),B}$ for which $F_{l(\boldsymbol{Z}),B} = true$ is predictive for the positive, or for the negative class. This decision is made by computing the information gain $ig(F_{l(\boldsymbol{Z}),B})$ for each singleton set $B = \{\boldsymbol{c}_{new}\}$, and summing the information gain values separately for tuples associated with the positive and negative class (lines 5.-12.). The obtained sums serve as a heuristic decision criterion for whether we proceed with a construction of a set B associated with positive or negative examples (lines 13.,16.). In either case, the construction of B is a greedy process, adding one candidate tuple at a time (lines 14.,15.).

4 Related Work

Most closely related to the RIG score in terms of purpose and applicability are the already mentioned lookahead strategies for relational learners.

The most basic form of lookahead is an exhaustive fixed-depth lookahead, where all possible refinements consisting of d or fewer literals are scored for their direct informativeness. Due to the exponential search space explosion, this approach is typically not feasible in practice even for small d. To alleviate the problem, one can reduce the search space by imposing user-defined constraints on admissible multi-literal refinements [2], or use purely syntactic conditions on admissible refinement *macros* [4]. Seeing that even depth-1 lookahead can be computationally very costly, Struyf et al. [18] propose an efficient approximation to exhaustive depth-1 lookahead, called feature based estimation (FBE). In FBE the space of admissible single-literal refinements is reduced, and information needed for scoring the candidate literals is pre-computed. None of these approaches tries to score the potential informativeness of a candidate literal without explicitly computing direct information scores obtainable by further refinement steps. A distinguishing feature of RIG score is that it measures the potential informativeness of a single-literal VI-refinement only based on the current refinement scenario consisting of the pseudo-iid example set and current query, but independent of all relations in the data not yet occurring in the query. The downside of this approach is that it can lead to overly optimistic estimates of potential informativeness, since the maximization over B sets in (8) and (9) is based on the implicit expectation that the data contains sufficiently many and informative attributes and relations that will allow to characterize the elements in B in subsequent refinement steps.

Also related to RIG scoring is the *ACORA* system described by Perlich and Provost [14]. This system constructs features for relational learning by measuring to what extent positive and negative examples are connected to different entities via a given candidate chain of relations. This is related to the RIG approach in that the identities of entities reached by a relation are considered, and a relation is considered informative if positive and negative examples are connected to different entities. In a crucial difference to the RIG approach, ACORA directly uses this kind of informativeness to construct features that refer to the identities of the objects reached by the chain. RIG, in contrast, is based on the assumption that object identifiers are not available for feature construction, and that therefore objects associated with negative and positive examples, respectively, need to be indirectly characterized by their attributes. The ACORA approach, thus, is applicable only when test and training examples are connected to the same set of potentially related objects (as e.g., in collaborative filtering scenarios, where test and training customers are connected to the same set of books that they might potentially be interested in). RIG, on the other hand, is applicable for the construction of models for domains that are not "stationary" in this sense (in molecular data, for example, no atom in a training molecule is connected to the same atoms as any atom in a test molecule).

5 Implementation details

5.1 RIG-TILDE

TILDE is a popular ILP system for learning logical decision trees in a top-down greedy fashion inspired by the propositional decision tree learner C4.5. We modified the algorithm by simply replacing the scoring heuristic with RIG^{exists} whenever the refinement $l(\boldsymbol{Z})$ introduces new variables, and standard information gain otherwise. The default scoring heuristic in TILDE is actually gain *ratio*, obtained by dividing information gain by the entropy according to the outcomes of the test (instead of the label). This splitting information should de-emphasize tests evenly spreading examples and is especially effective for conditions hav-

ing a large number of possible outcomes. While it is straightforward to conceive a ratio version for both RIG^{exists} and RIG^{count}, it is not necessarily the case that a large splitting information implies a lower *potential* informativeness: even in the extreme case of a one-to-one mapping between training examples $\boldsymbol{c}$ and bindings $\boldsymbol{c}_{new}$—a totally useless attribute in itself—entities $\boldsymbol{Z}_{new}$ could be easier to discriminate by further refinements.

In order to compute RIG^{exists}, the set of candidate constants to be considered for addition to the set B needs to be generated. Given a refinement scenario $dv(\boldsymbol{X}), Q(\boldsymbol{Y})$, which corresponds to a node in the tree being constructed, and a candidate refinement $l(\boldsymbol{Z})$, we need to compute the set:

$$\{\boldsymbol{c}_{new} \in \mathcal{C}^{\boldsymbol{Z}_{new}} \mid \mathcal{D} \models \exists \boldsymbol{Y} \setminus \boldsymbol{X} \ : Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\boldsymbol{X}/\boldsymbol{c}, \boldsymbol{Z}_{new}/\boldsymbol{c}_{new}]\} \tag{10}$$

for each training example $\boldsymbol{c} \in dv(\boldsymbol{X})$ (cf Table 3.1.2, line 4.). Plain TILDE, on the other side, simply needs to verify whether this set is non-empty:

$$\exists \boldsymbol{c}_{new} \in \mathcal{C}^{\boldsymbol{Z}_{new}} : \mathcal{D} \models \exists \boldsymbol{Y} \setminus \boldsymbol{X} \ : Q(\boldsymbol{Y})[\boldsymbol{X}/\boldsymbol{c}], l(\boldsymbol{Z})[\boldsymbol{X}/\boldsymbol{c}, \boldsymbol{Z}_{new}/\boldsymbol{c}_{new}] \tag{11}$$

Equation 10 was implemented with the following Prolog query:

$$\texttt{setof}(\boldsymbol{Z}_{new}, (\boldsymbol{Y} \setminus \boldsymbol{X})\hat{}(Q(\boldsymbol{Y}), l(\boldsymbol{Z}))[\boldsymbol{X}/\boldsymbol{c}], \boldsymbol{S}) \tag{12}$$

where $\boldsymbol{S}$ collects the set of possible bindings.

To avoid the problem described in section 3.1.1 of introducing multiple equivalent copies of the same literal, we force the next refinement of $Q(\boldsymbol{Y}), l(\boldsymbol{Z})$ to only use literals with at least one variable from $\boldsymbol{Z}_{new}$. To prevent too long clauses, we also bound the maximum number of variables $|\boldsymbol{Y} \cup \boldsymbol{Z}|$ which can appear in the query.

Finally, it is worthwhile noting that the post-pruning feature implemented in TILDE has an additional advantage in the RIG-TILDE version: whenever the potential informativeness, which guided the selection of a certain relational refinement, does not eventually materialize further down in the search, the unlucky guess will be pruned away in the post-processing phase.

5.2 WRIG-FOIL

The definition of RIG in section 3 is based on two major components: (1) it was obtained as a standard information gain from a suitably defined new attribute in a pseudo-iid data view and (2) it measures potential informativeness by considering the informativeness of subsets B of possible bindings for the new variables.

When implementing RIG as a replacement for the native *weighted information gain (WIG)* scoring heuristic in the FOIL system, it turns out that the first design principle of RIG conflicts with the FOIL architecture: as a standard information gain measure, RIG is symmetric in how it treats positive and negative examples. Conditional class distributions with a high posterior probability for the negative class are rated just as highly as posteriors with a high probability for the positive class. Moreover, when conditioning on some feature F, both conditionals $P(\cdot \mid F = true)$ and $P(\cdot \mid F = false)$ influence the score in the same way. The FOIL architecture conflicts with these symmetries in two ways: first, FOIL iteratively covers positive examples, and can only make use of an increased posterior probability for the positive class. Second, examples not covered by a current refinement are discarded (for the construction of the current clause), and therefore a reduction in the conditional entropy $P(\cdot \mid F = false)$ is of no interest. For these reasons we have implemented in FOIL a

modification of the RIG scoring metric that is better adapted to the FOIL architecture, and combines elements of WIG and RIG score:

$$\begin{aligned}&\mathit{WRIG}(l(\boldsymbol{Z})) := \\ &\quad \max_{B \subseteq \mathcal{C}^{\boldsymbol{Z}_{new}}} \Big(P(F^{\mathit{exists}}_{l(\boldsymbol{Z}),B} = \mathit{true}) \cdot \\ &\qquad\qquad (-\log(P(\mathit{class} = \mathit{true})) + \log(P(\mathit{class} = \mathit{true} \mid F^{\mathit{exists}}_{l(\boldsymbol{Z}),B} = \mathit{true}))) \Big)\end{aligned}$$

The WRIG measure borrows three main elements from WIG: the score depends only on the change of $-\log(P(\mathit{class} = \mathit{true}))$ as a measure of the proportion of positive examples. When considering a VI-refinement $l(\boldsymbol{Z})$, only those examples that have an extension in the new local training set defined by the refinement are considered (in our setup, these are the examples for which $F^{\mathit{exists}}_{l(\boldsymbol{Z}),B} = \mathit{true}$). Finally, the change in class purity is weighted with the factor $P(F^{\mathit{exists}}_{l(\boldsymbol{Z}),B} = \mathit{true})$ of how many examples of the original training set are represented in the new local training set. Two key elements are taken from RIG: potential informativeness is measured by considering and maximizing over the B-relativized features $F^{\mathit{exists}}_{l(\boldsymbol{Z}),B}$. Second, $P(\mathit{class} = \mathit{true} \mid F^{\mathit{exists}}_{l(\boldsymbol{Z}),B} = \mathit{true})$ measures the proportion of positive examples still in the original training set, not (as WIG) in the extended training set obtained from the refinement $l(\boldsymbol{Z})$. WRIG, unlike WIG, thus does not produce high scores just because positive examples tend to have more extensions with $l(\boldsymbol{Z})$ than negative examples (a behavior of the score function that does not suit FOIL's existential semantics for new variables very well).

The implementation of WRIG within FOIL system is hence straightforward. When building a clause, WRIG is used for the scoring refinements. In the case of basic refinements, this is equivalent to using FOIL's original WIG measure.

The computation of WRIG can be performed using the same target tables which FOIL generates as local training sets when computing weighted information gain: the computational cost is then linear in the dimension of such target tables, and hence there are no fundamental complexity differences between FOIL and WRIG-FOIL.

6 Experiments

6.1 TILDE experiments

6.1.1 Synthetic data

We use synthetic *slotchain* data [8] to test RIG's ability to identify potentially informative literals. This data set is a larger and more elaborate version of the kind of structure shown in Figure 1. In this data, an example X is *positive*, if and only if an entity Z with $\mathit{att}(Z) = \mathit{true}$ can be reached via the chain of relations $r_{0,0}, r_{1,0}, r_{2,0}, r_{3,0}$. Thus, the target clause to find in this data is

$$\mathit{positive}(X) \leftarrow r_{0,0}(X, Y_1), r_{1,0}(Y_1, Y_2), r_{2,0}(Y_2, Y_3), r_{3,0}(Y_3, Z), att(Z). \qquad (13)$$

The $r_{i,0}$-literals are neither directly informative nor determinate. The data set consists of approximately 5,400 true ground facts and also includes "noise relations" $r_{i,j}$ ($i = 0, \ldots, 3$,

$j = 1, \ldots, 2$) that have no predictive value. Standard TILDE is able to recover the target clause (13) only using an exhaustive lookahead of 4, otherwise the gain ratio scoring measure used by TILDE is not able to capture the potential informativeness of the literals composing the slotchain. Being limited to depth-1 lookahead, the FBE algorithm [18], described in section 4, also is unable to retrieve the target slotchain clause. The total number of Prolog queries required by Tilde-L4 to recover the slotchain is 31,125. RIG-TILDE is instead able to recover the target clause from data, without the use of lookahead, with only 43 queries.

6.1.2 Struyf et al.'s data sets

We report here about the applicability and the performance of RIG-TILDE in the data sets employed in [18], i.e., Mutagenesis (Muta188 and Muta230), Financial, Sisyphus, Carcinogenesis, University of Washington (UWCSE), Yeast, and Bongard. For details on the data sets see [18] and references therein. Results obtained by using the same 10-fold cross validation as in [18] are shown in Table 5. We report accuracy, area under the precision-recall curve, the number of Prolog queries which are evaluated throughout the search and the number of literals in the final (pruned) tree, both averaged per fold.

6.1.3 Activities of daily living

Activities of daily living (ADL) [12] is an activity recognition data set which describes the activities of a user having breakfast at home. The data consists in observations of 4,597 tagged events divided in 20 time sequences, where the user performs activities such as reading the newspaper or toasting bread (19 total activities, including *nil*) and interacts with several objects, such as kettle, teabag, butter (23 total objects). For each time sequence, interactions between user and objects are observed (data have been obtained using RFID readers), and the duration of each activity is known as well. No additional background knowledge is used. We extracted three different binary classification tasks: ADL_1 (any activity vs. no activity), ADL_2 (*readNewspaper* vs. rest), and ADL_3 (*makeCereals_eat* vs. rest). As in [12], a leave-one-sequence-out approach was used for the experiments. We show in Table 6 the F_1 measure rather than accuracy, owing to the unbalanced nature of the data set. We also report the number of evaluated refinements, the number of literals in the final tree and the computational time.

Following the approach by Struyf et al. [18], propositional attributes were encoded using relations of the form *Attribute$_j$(id,value)*: for this reason, TILDE with no lookahead cannot properly retrieve informative literals (results not reported).

6.1.4 Gene essentiality

This is a crucial problem in cellular biology, which can help to understand the minimal requirements of cellular life, as well as to develop new drugs. The goal is to predict whether a certain gene is essential for the life of the cell (a binary classification task). Machine learning algorithms for solving it have been recently studied in the literature [9, 1].

We started from the two propositional data sets used in [9]. The *s.cerevisiae* (yeast) data set contains 4,728 genes (967 essential and 3,762 nonessential) described by 42 attributes. The first 16 attributes (that include e.g., phyletic retention, number of paralagous genes, aromaticity score, amino acid composition) are obtainable from sequence data alone while the

Table 5 Comparison between TILDE with lookahead $L = 0, 1, 2$, FBE and RIG-TILDE on Struyf et al.'s data sets. Evaluated refinements, the number of literals in the final tree, and computational time are averaged per fold. Significant wins/losses of Tilde-Lx and FBE with respect to RIG are indicated by $\oplus$ and $\ominus$, respectively (p-value < 0.01, paired t-test).

Data set	Method	Accuracy		AUPRC		Evaluated refinements	Literals in final tree	Time (s)
Muta188	L0	69.1±7.5		70±8	$\ominus$	168	1	0.0
	L1	74.5±4.7		84±7		50,880	26	28.5
	L2	73.9±6.7		79±6		517,383	39	689.9
	FBE	74.3±8.6		85±8		5,253	28	2.9
	RIG	72.8±8.3		84±10		15,388	13	41.7
Muta230	L0	63.9±3.3		65±4		179	2	0.0
	L1	74.8±5.8		84±3		178,867	38	5.0
	L2	73.5±3.4		81±7		1,046,051	46	167.4
	FBE	75.4±7.6		86±4		9,734	37	1.5
	RIG	68.9±8.8		68±17		11,309	14	43.0
Financial	L0	86.8±0.7	$\ominus$	13±1	$\ominus$	28	0	0.0
	L1	96.6±1.5		84±9		5,565	3	1.7
	L2	96.2±1.8		81±9		71,417	8	461.9
	FBE	96.6±2.5		84±9		510	3	1.7
	RIG	95.8±1.2		90±12		1,195	16	22.9
Carcinog	L0	62.1±4.5		66±4		14,283	15	1.4
	L1	60.3±4.1		67±4		359,920	79	184.2
	L2	60.0±3.4		64±4		2,596,396	165	35,841.2
	FBE	60.0±7.6		67±5		29,155	63	34.1
	RIG	61.8±1.8		64±7		11,710	18	13.3
UWCSE	L0	93.6±2.3		39±17		6,192	25	2.4
	L1	94.0±2.3		29±14		253,425	113	71.9
	L2	94.3±2.3		33±13		2,041,260	101	3,586.0
	FBE	94.8±1.0		34±19		14,604	68	11.9
	RIG	95.2±0.8		41±26		5,873	18	75.1
Yeast	L0	87.7±0.4		68±2		399,203	91	50.7
	L1	88.0±0.6		63±2		2,909,296	168	401.5
	L2	88.0±0.5		62±2		92,638,421	154	16,708.5
	FBE	88.7±0.7		71±1		527,758	154	106.6
	RIG	87.6±0.8		67±4		240,516	91	366.6
Bongard	L0	98.1±0.4		98±1	$\ominus$	2,404	11	6.1
	L1	99.6±0.3	$\oplus$	100±0	$\oplus$	10,399	19	15.2
	L2	100.0±0.0	$\oplus$	100±0	$\oplus$	86,072	13	1,595.7
	FBE	99.5±0.8		100±0		589	17	3.3
	RIG	97.6±0.8		99±1		696	18	19.1
Sisyphus A	L0	62.1±0.0	$\ominus$	62±0	$\ominus$	40	0	0.4
	L1	94.9±0.5		97±1		652,634	36	297.1
	L2	96.6±0.2	$\oplus$	98±0		1,621,575	66	14,140.2
	FBE	94.8±0.3		97±1		16,651	34	54.8
	RIG	95.5±0.4		97±1		13,850	42	1,598.1
Sisyphus B	L0	71.4±0.0	$\ominus$	29±0	$\ominus$	2	0	0.2
	L1	75.9±0.7	$\ominus$	59±1	$\ominus$	1,458,210	115	286.9
	L2	92.0±0.3	$\oplus$	86±1		1,886,339	32	18,255.0
	FBE	76.1±0.7	$\ominus$	59±2	$\ominus$	36,192	68	12.0
	RIG	81.7±1.6		75±4		55,947	127	1,852.6

Table 6 Results on activities of daily living data. F_1 is the micro-average on the 20 sequences. Evaluated refinements, the number of literals in the final tree and computational time are averaged per sequence.

Data set	Method	F_1	AUPRC	Evaluated refinements	Literals in final tree	Time (s)
ADL_1	L1	83.5	45.8±36.0	288,036	30	24.4
	L2	82.6	47.0±33.1	1,449,520	39	220.1
	FBE	84.4	39.7±31.6	18,848	110	3.9
	RIG	84.6	66.2±26.2	3,180	22	5.4
ADL_2	L1	88.2	90.8±10.0	50,791	36	4.5
	L2	87.4	89.3±13.0	401,738	52	151.7
	FBE	88.5	91.0±10.2	4,537	30	1.5
	RIG	88.5	89.6±9.6	878	27	2.3
ADL_3	L1	80.4	86.1±12.5	23,734	153	2.0
	L2	74.4	76.7±16.6	506,127	157	69.0
	FBE	80.4	86.1±12.5	3,754	35	1.4
	RIG	80.6	81.4±12.2	950	87	2.9

other 26 attributes (that include number of interacting proteins, or subcellular localization) require extensive wet laboratory work. For our experiments, we only considered the first set of attributes. The *e.coli* data set contains 3,570 genes (612 essential and 2,958 nonessential) each described by 28 attributes obtainable from sequence only. Continuous attributes were discretized using the entropy minimization heuristic [7] as in [9].

We then enriched both data sets with relational information consisting of protein-protein interactions derived from the STRING data base [10]. Since the sources of evidence for association between proteins can be very noisy, we retained only pairs with an interaction score above 0.9.

For the sake of comparison, we replicated the bootstrapping evaluation procedure reported in [9]: data was split maintaining 50% of the examples of each class, both in the training and in the test set. The procedure was then repeated n times and final gene essentiality probabilities were obtained as the averages of the probabilities assigned in each trial. When reproducing results in [9] we observed no significant advantages in using more than 10 repetitions so we report results with n = 10 rather than n = 100 as in [9].

As in the ADL data sets, propositional attributes were represented by relations so TILDE with no lookahead cannot properly retrieve informative literals (results not reported).

Following the approach in [9], we show in Table 7 the precision obtained when predicting as positives the top n% of the genes ordered by predicted probability, with $n = 1, 5, 10, 15, 20$. Since TILDE-L2 ran for over 24 hours on a single train/test split without terminating, we compare only to TILDE-L1 on this data set. FBE results are also not reported, because the system crashed when run on this data set. Both RIG-TILDE and Tilde-L1 achieve state-of-the-art results for this task.

6.2 FOIL experiments

6.2.1 Synthetic data

We performed also with WRIG-FOIL the same experiment on Slotchain data set described for RIG-TILDE: the use of WRIG is again decisive in order to correctly retrieve the target clause (13), which plain FOIL is otherwise not able to find. WRIG-FOIL was also run

Table 7 Results on protein essentiality data. Positive predicted value at top n% predictions is reported. Evaluated refinements, the number of literals in the final tree, and computational time are averaged per run.

Data set	Method	Top 1%	Top 5%	Top 10%	Top 15%	Top 20%	Evaluated refinements	Literals in final tree	Time (s)
s.cerev	L1	62	72	64	59	54	370,431	30	119.7
	RIG	79	74	64	58	53	2,275	38	48.4
	ref. [9]	81	64	57	54	49	-	-	-
e.coli	L1	89	72	60	50	43	572,530	57	70.1
	RIG	80	73	61	53	45	7,835	66	114.8
	ref. [9]	82	68	58	48	40	-	-	-

in a second setting, without performing the maximization over the B set (called WRIG-FOIL$_{noLA}$). This modified version of our algorithm does not take advantage of the "lookahead" capacity of WRIG—but, differently from FOIL, employs the counting of examples rather than substitutions—and it is therefore not able to retrieve the slotchain target clause.

6.2.2 Real data

Table 8 Results on real world data sets using plain FOIL, WRIG-FOIL and WRIG-FOIL$_{noLA}$. F_1 is the micro-average over the 10 folds. The accuracy, the number of learned clauses, the number of evaluated refinements, and the computational time are averaged per fold.

	Method	Accuracy	F_1	Learned clauses	Evaluated refinements	Time (s)
Carcinogenesis	FOIL	51.5±6.4	52.9	9.9	23,036	77.1
	WRIG-FOIL	56.1±8.3	56.2	10.3	73,206	1779.7
	WRIG-FOIL$_{noLA}$	50.9±5.9	42.6	9.5	8,845	49.1
Muta188	FOIL	66.0±10.4	74.4	8.4	53,709	200.2
	WRIG-FOIL	74.0±9.7	81.1	7.2	10,267	97.1
	WRIG-FOIL$_{noLA}$	78.8±8.3	84.6	6.4	6,706	72.6
Muta230	FOIL	62.3±7.0	68.1	7.2	80,835	680.5
	WRIG-FOIL	68.3±8.7	73.8	8.4	27,053	1165.5
	WRIG-FOIL$_{noLA}$	74.6±5.7	80.1	5.8	7,273	101.3
UWCSE	FOIL	95.3±0.8	13.6	3.4	6,394	9.5
	WRIG-FOIL	94.5±2.0	18.7	2.8	3,005	5.0
	WRIG-FOIL$_{noLA}$	94.8±1.4	19.1	2.8	2,363	1.3

For FOIL experiments we employed the Carcinogenesis, Mutagenesis and UWCSE data sets, which were used also for RIG-TILDE. For all the tasks, FOIL's threshold for the minimum acceptable accuracy of a rule has been set to 50%, and negated literals were forbidden. A 10-fold cross-validation was performed on each data set, except for UWCSE, where we used the leave-one-area-out setting. Results obtained are shown in Table 8.

6.3 Discussion

The experiments performed with TILDE show that the use of RIG greatly lowers the number of queries being evaluated throughout the search, with respect to lookahead: RIG-TILDE evaluates a number of queries which is usually of the same order of magnitude of TILDE

without lookahead, and about 1% (from 0.6% for *s.cerev* to 20% for Financial) and 0.5% (from 0.2% for ADL_1 to 20% for Financial) of the ones evaluated by TILDE-L1 and -L2, respectively. Note that our savings in the number of evaluated queries does not directly translate into faster learning, because of the difference between computing (10) and (11) which is highly problem-dependent. Using the naive implementation reported in (12), we obtained comparable computing times on average between RIG-TILDE and TILDE-L1, whereas TILDE-L2 was generally one or two orders of magnitude slower.

RIG-TILDE achieves results which are on average comparable to the best ones obtained by varying the amount of lookahead, while generating simpler models in terms of number of literals in most cases.

The gene essentiality task shows that the use of RIG can be of great impact also in real-world complex problems, in which the use of deep lookahead can be very expensive, and sometimes prohibitive. The relational approach in this data set outperforms results in [9] that are based on a simple propositional Naive Bayes.

The additional FOIL experiments demonstrate the applicability of RIG scoring in different learning systems. However, since the new WRIG score differs from the native FOIL scoring function also in other aspects than RIG's implicit lookahead capabilities, it is not directly clear to what extent the observed improvement of WRIG-FOIL over FOIL is due to this main novel feature of WRIG. Experiments with the modified version WRIG-FOIL_{noLA} provide inconclusive results, with only the experiments on Carcinogenesis indicating a major impact of the lookahead feature.

7 Conclusions

Relational information gain is a refinement scoring function that has a sound information-theoretic justification. We have introduced an algorithm for calculating approximate RIG scores and implemented it in conjunction with two popular ILP systems that use literal scoring heuristics: TILDE and FOIL. RIG, however, is not specifically conceived for these systems and its scope and applicability is more general. In conjunction with both learners, our experiments on synthetic slotchain data clearly show the ability of RIG in discovering potential informativeness of a literal, without requiring a lookahead. In the experiments with real data RIG was competitive in terms of accuracy and speed with other state of the art methods. However, no evidence was found that long slotchain like dependencies played a major role for the prediction tasks in these datasets. We can therefore conclude that RIG scoring is a widely applicable approach that is generally viable, with the potential of giving superior results in domains that are characterized by probabilistic dependencies transmitted over chains of several relations.

8 Acknowledgments

The authors would like to thank Hendrik Blockeel for providing us the TILDE system, Jan Struyf for his valuable help with TILDE experiments, and the anonymous reviewers for their insightful comments.

References

1. C. Allauzen, M. Mohri, and A. Talwalkar. Sequence kernels for predicting protein essentiality. In *Proceedings of the 25th international conference on Machine learning*, pages 9–16. ACM New York, NY, USA, 2008.
2. H. Blockeel and L. De Raedt. Lookahead and discretization in ILP. In *Proc. of the 7th Int. Workshop on ILP*, pages 77–84, 1997.
3. H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1–2):285–297, 1998.
4. L. P. Castillo and S. Wrobel. A comparative study on methods for reducing myopia of hill-climbing search in multirelational learning. In *Proc. of the 21st Int. Conf. on Machine Learning*, 2004.
5. L. De Raedt. Logical settings for concept-learning. *Artificial Intelligence*, 95(1):187 – 201, 1997.
6. L. De Raedt. *Logical and Relational Learning*. Springer, 2008.
7. U.M. Fayyad and K.B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(1):87–102, 1992.
8. P. Frasconi, M. Jaeger, and A. Passerini. Feature discovery with type extension trees. In *Proc. of the 18th Int. Conf. on Inductive Logic Programming*, pages 122–139, 2008.
9. A. M. Gustafson, E. S. Snitkin, S. C. J. Parker, C. DeLisi, and S. Kasif. Towards the identification of essential genes using targeted genome sequencing and comparative analysis. *BMC Genomics*, 2006.
10. L.J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, et al. STRING 8–a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Research*, 37(Database issue):D412, 2009.
11. A. J. Knobbe, A. Siebes, and D. van der Wallen. Multi-relational decision tree induction. In *Proceedings of PKDD-99*, pages 378–383, 1999.
12. N. Landwehr, B. Gutmann, I. Thon, L. De Raedt, and M. Philipose. Relational transformation-based tagging for activity recognition. *Fundam. Inform.*, 89(1):111–129, 2008.
13. J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of SIGKDD'03*, 2003.
14. C. Perlich and F. Provost. Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, 62:65–105, 2006.
15. J.R. Quinlan. Determinate literals in inductive logic programming. In J. Mylopoulos and R. Reiter, editors, *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence*, 1991.
16. J.R. Quinlan and R.M. Cameron-Jones. FOIL: A midterm report. In *European Conference on Machine Learning*, page 3. Springer, 1993.
17. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1–2):107–136, 2006.
18. J. Struyf, J. Davis, and D. Page. An efficient approximation to lookahead in relational learners. In *Proceedings of ECML-06*, volume 4212 of *LNAI*, pages 775–782, 2006.
19. A. Van Assche, C. Vens, H. Blockeel, and S. Dzeroski. First order random forests: Learning relational classifiers with complex aggregates. *Machine Learning*, 64:149–182, 2006.