Change of Representation for Statistical Relational Learning

Jesse Davis, Irene Ong, Jan Struyf, Elizabeth Burnside David Page
University of Wisconsin - Madison
1210 West Dayton
Madison, WI 53706, USA
email: jdavis@cs.wisc.edu

Vítor Santos Costa
Universidade Federal do Rio de Janeiro
Centro de Tecnologia, Bloco H-319
Rio de Janiero, Brasil

Abstract

Statistical relational learning (SRL) algorithms learn statistical models from relational data, such as that stored in a relational database. We previously introduced *view learning* for SRL, in which the view of a relational database can be automatically modified, yielding more accurate statistical models. The present paper presents SAYU-VISTA, an algorithm which advances beyond the initial view learning approach in three ways. First, it learns views that introduce new relational *tables*, rather than merely new fields for an existing table of the database. Second, new tables or new fields are not limited to being approximations to some target concept; instead, the new approach performs a type of predicate invention. The new approach avoids the classical problem with predicate invention, of learning many useless predicates, by keeping only new fields or tables (i.e., new predicates) that immediately improve the performance of the statistical model. Third, retained fields or tables can then be used in the definitions of further new fields or tables. We evaluate the new view learning approach on three relational classification tasks.

1 Introduction

Most statistical relational learning (SRL) algorithms are constrained to operate with the specific representation they are given—a PRM [Friedman *et al.*, 1999] must use the schema of the input database, a logic-based system must use the predicates provided. Yet, in many cases the original relational representation was not designed to empower the learning algorithm. For example, the schema for a medical database may have been chosen to simplify billing; biological databases often have a hierarchical schema, forcing an SRL system to use long slot-chains or long clause bodies to find related data.

In some cases an SRL user may have the freedom to modify the representation for the sake of learning. Even so, considering all relevant features or relations to include for a task is a difficult job by itself. Ideally, we would like the learning algorithm to be able to discover and incorporate relevant, intermediate concepts into the representation. For instance, consider the well-known task of predicting whether two citations refer to the same underlying paper. The *CoAuthor* relation is potentially useful for disambiguating citations; for example, if S. Russell and S.J. Russell both have similar lists of coauthors, then perhaps they are interchangeable in citations. But the *CoAuthor* relation may not have been provided to the learning system. Furthermore, *CoAuthor* can be used as a building block to construct further explicit features for the system, such as a new predicate *SamePerson*.

A start already has been made in change of representation for SRL systems, under the name of *View Learning*. Initial approaches to View Learning [Davis *et al.*, 2005b; 2005a] showed that even an application with a single table can benefit from new views: views can represent connections between related examples. Moreover, a greedy approach to view learning can be useful. The present paper introduces important extensions for change of representations in SRL. First, it provides a mechanism for learning a new view as a full *new relational table*, such as *CoAuthor*. Second, it permits *a newly-invented relation, or predicate, to be used in the invention of other new relations*, such as *SamePerson*. Such re-use goes beyond simply introducing "short-cuts" in the search space for new relations; because the new approach also permits a relation to be from aggregates over existing relations, re-use actually extends the space of possible relations that can be learned by the approach. Because this new work extends SAYU by providing a mechanism for View Invention by Scoring TAbles, we call the resulting system SAYU-VISTA.

2 ILP and SAYU

Inductive logic programming (ILP) is a popular approach for learning in a relational environment. Given a set of positive and negative examples and background knowledge, an ILP system finds a logical description of the underlying data model that differentiates between the positive and negative examples. This description is a set of first-order logical *rules* or clauses, which form a logic program.

SAYU is an SRL system that combines ILP with Bayesian network learning, by default with tree-augmented naive Bayes learning [Friedman *et al.*, 1997]. SAYU is an acronym for "Score As You Use." Unlike other approaches to using ILP for defining new features, SAYU scores each clause not by a standard ILP measure, but by how much it helps the

model in which the clause is used. Another system, known as nFOIL, was developed in parallel with SAYU and has this same property [Landwehr *et al.*, 2005]. In the SAYU approach, we start from an empty model (or a prior model). Next, an ILP system generates rules. Each generated rule represents a new feature which is added to the current model. We then evaluate the generalization ability of the model extended with the new feature, where generalization ability is measured as the area under the precision recall curve on a held-aside subset of the data. We retain the new model if the inclusion of the new feature significantly improves the model's generalization ability; otherwise we remain with the original model. This results in a tight coupling between feature construction and model building.

SAYU needs an ILP system to propose rules. In our work, we use Aleph, which implements the Progol algorithm [Muggleton, 1995] to learn rules. This algorithm induces rules in two steps. Initially, it selects a positive instance to serve as the "seed" example. It searches the background knowledge for the facts known to be true about the seed example. The combination of these facts forms the example's most specific or saturated clause. The key insight of the Progol algorithm is that some of these facts explain this example's classification. Thus, generalizations of those facts could apply to other examples. Aleph defines the search space to be clauses that generalize a seed example's saturated clause, and performs a general to specific search over this space.

SAYU greedily searches for new fields or variables, defined by first-order logic clauses, that improve the prediction of the class field. SAYU modifies the standard Aleph search as follows. Instead of using coverage, Aleph passes each clause it constructs to SAYU, which converts it to a binary feature. The feature is added to the current training set and SAYU learns a new Bayes net, a TAN network in our case, incorporating this new feature. We measure performance by looking at the area under the precision recall curve on a held side tune set. If the feature degrades the performance of the network, SAYU discards the feature and reverts back to the old classifier. Then SAYU returns control to Aleph to construct the next clause. If the new feature improves the score of the network, then SAYU retains the feature in the network. In contrast to Aleph, after accepting a rule, SAYU randomly selects a new seed example and reinitializes the search. Thus, for a given seed, SAYU does not search for the best rule, but only the first rule that helps. However, nothing prevents the same seed from being selected multiple times during the search.

3 Learning New Predicates

The initial approach to View Learning [Davis *et al.*, 2005b], suffers from two important drawbacks. First, it only creates new fields, not new tables. The new field definition has the same arity as the target predicate. Second, the new fields are just learned approximations to the target concept. SAYU-VISTA addresses both shortcomings. It creates new predicates with arity greater than one, some capturing many-to-many relations, which require a new table to represent. Furthermore these new predicates are no longer approximations to the target concept, but may be any concept that improves the statistical model.

The original motivation for view learning centered on learning a statistical expert system to provide decision support to radiologists [Davis *et al.*, 2005b]. There we used SRL because the learned statistical model sits on top of the National Mammography Database (NMD) schema, a standard established by the American College of Radiology [ACR, 2004]. The goal of the data set is to predict which abnormalities on a mammogram are malignant. We will use mammography as a running example to help illustrate the key components of the algorithm.

SAYU-VISTA, nFOIL and SAYU all learn definite clauses and evaluate clauses by how much they improve the statistical classifier. The key difference in the algorithms rests in the form that the head of the learned clauses takes. In nFOIL and SAYU, the head of a clause has the same arity and type as the example, allowing us to precisely define whether a clause succeeds for a given example and hence whether the corresponding variable is true. In the Mammography domain, a positive example has the form `malignant(ab1)`, where `ab1` is a primary key for some abnormality. Every learned rule has the head `malignant(A)` such as in the following rule:

```
malignant(Ab1) if:
   ArchDistortion(Ab1,present),
   same_study(Ab1,Ab2),
   Calc_FineLinear(Ab2,present).
```

The Bayesian network variable corresponding to this rule will take value *true* for the example `malignant(ab1)` if the clause body succeeds when the logical variable `A` is bound to `ab1`.

SAYU-VISTA removes the restriction that all the learned clauses have the same head. First, SAYU-VISTA learns predicates that have a higher-arity than the target predicate. For example, in the Mammography domain, predicates such as `p11(Abnormality1, Abnormality2)`, which relate pairs of abnormalities, are learned. Subsection 3.1 discusses scoring predicates that have higher arities than the target relation. Second, SAYU-VISTA learns predicates that have types other than the example key in the predicate head. For example, a predicate `p12(Visit)`, which refers to attributes recorded once per a patient visit, could be learned. In order to score predicates of this form, we introduce the concept of *Linkages*, which are discussed in subsection 3.2. After discussing how to evaluate these types of predicates, we will present the full SAYU-VISTA algorithm.

3.1 Scoring Higher Arity Predicates

SAYU-VISTA can learn a clause such as:

```
p11(Ab1,Ab2) if:
   density(Ab1,D1),
   prior-abnormality-same-loc(Ab1,Ab2),
   density(Ab2,D2),
   D1 > D2.
```

This rule says that *p11* is true of a pair of abnormalities *Ab1* and *Ab2* if they are at the same location, *Ab1* was observed first, and *Ab2* has higher density than *Ab1*. Thus

p11 may be thought of as "density increase." Unfortunately, it is not entirely clear how to match an example, such as `malignant(ab1)`, to the head of this clause for *p11*. SAYU-VISTA maps, or links, one argument to the example key and aggregates away any remaining arguments using existence or count aggregation. The next section describes the approach used for linkage; the remainder of this paragraph discusses aggregation. In existence aggregation, the clause succeeds for the given example (key) if there exist *any* bindings of the remaining variables for which the clause succeeds. Count aggregation computes the *number* of bindings for these remaining variables for which the clause succeeds. Currently, SAYU-VISTA discretizes aggregated features using a binning strategy that creates three equal-cardinality bins, where three was chosen arbitrarily before the running of any experiments.

3.2 Linkages

So far we have simplified matters by assuming that the first argument to the learned predicate has the same type as the example key. In our examples so far, this type has been *abnormality id*. There is no need to enforce this limitation. For example, in predicting whether an abnormality is malignant, it might be useful to use the following clause, where `Visit` is a key that refers to all abnormalities found on a given mammogram:

```
p(Visit) :-
   visit(Visit,Ab),
   MassesShape(Ab,oval).
```

Predicate `p` is true of a visit, or mammogram, that contains at least one abnormality with an oval shape.

Linkage declarations are background knowledge that can establish the connection between objects in the examples and objects in the newly invented predicates. When these objects are of the same type, the linkage is trivial; otherwise, it must be defined. For mammography, we use linkage definitions that link an *abnormality* to its *patient* or to its *visit* (mammogram). The linkages for the other datasets we use are equally straightforward and are presented when we describe those datasets.

3.3 Predicate Learning Algorithm

At a high level SAYU-VISTA learns new predicates by performing a search over the bodies of definite clauses and selecting those bodies that improve the performance of the statistical model on a classification task. We use tree-augmented naive Bayes (TAN) [Friedman *et al.*, 1997] as our statistical model.

The predicate invention algorithm takes several inputs from a user. First, it needs a training set, which is used to learn the statistical model, and a tuning set, which is used to evaluate the statistical model. The user provides a pre-defined set of distinguished types, which can appear in the head of a clause. The user provides background knowledge, which must include linkage definitions for each distinguished type. The algorithm using an improvement threshold, p, to decide which predicates to retain in the model. A new predicate must improve the model's performance by at least $p\%$ in order to be kept. We used $p = 0.02$ in all experiments. Optionally, the user may input an initial feature set to the algorithm. Algorithm 1 shows psuedocode for the SAYU-VISTA algorithm.

The clause search proceeds as follows. We randomly select an arity for the predicate. To limit the search space, we restrict the arity to be either the arity of the target relation, or the arity of the target relation plus one. Next, we randomly select the types for the variables that appear in the head of the clause. The clause search uses a top-down, breadth-first refinement search. We define the space of candidate literals to add using modes, as in Progol [Muggleton, 1995] or Aleph [Srinivasan, 2001]. We score each proposed clause by adding it as variable in the statistical model. To construct the feature, we first link the predicate back to the example key as described in subsection 3.2. Then we perform the necessary aggregation, discussed in subsection 3.1, to convert the clause into a feature. By default, the algorithm first tries existence aggregation and then tries count aggregation. The clause search *terminates* in three cases: **(i)** it finds a clause that meets the improvement threshold; **(ii)** it fully explores the search space; **(iii)** it exceeds the clause limit. After satisfying one of these conditions, the algorithm re-initializes the search process. Every clause that meets the improvement threshold is added into the background knolwedge. Therefore, future predicate definitions can re-use previously learned predicates. As in prior work [Davis *et al.*, 2005a], the algorithm terminates when it exceeds the global time limit.

4 Data and Methodology

Cora. The objective of this dataset is to predict whether two citations refer to the same paper. The dataset was originally constructed by McCallum et al. [McCallum *et al.*, 2000]. We used the same version of the data as Kok and Domingos [Kok and Domingos, 2005]. Cora includes 1295 citations to 112 Computer Science papers, resulting in 25072 positive examples and 597310 negative examples. The background knowledge includes data on title, venue, author(s), and year for each citation. We defined *paper*, *title*, *venue*, *author* and *year* as keys that can appear in heads of clauses. We link a paper to its title, venue, author(s) and year fields. We aggregate over papers and authors.

UW-CSE. This common SRL dataset was constructed by Richardson and Domingos [Richardson and Domingos, 2006] and is publicly available. The goal is to predict the advisor of a graduate student. The information comes from the University of Washington CS Department and contains 113 positive examples versus 2,711 negative examples. We defined *students*, *professors*, *courses* and *publications* as keys that could appear in the head of a clause. We link a course to a graduate student by the TA relationship, and we link papers to a graduate student by the author relationship. We link a course to a professor by the teaches relationship and we link papers to a professor by the author relationship. We aggregate over students, professors, papers and courses.

Mammography. The objective of this dataset is to predict whether an abnormality on a mammogram is benign or malignant [Davis *et al.*, 2005b]. This dataset consists of a radiologist's interpretation of a mammogram and not the raw image data. The dataset contains 435 positive examples and

Input: Train Set Labels T, Tune Set Labels S, Distinguished Types D, Background Knowledge B, Improvement Threshold p, Initial Feature Set F_{init}
Output: Feature Set F, Statistical Model M

```
F = F_init;
BestScore = 0;
while time remains do
    Randomly select the arity of predicate to invent;
    Randomly select types from D for each variable in the head of the predicate;
    SelectedFeature = false;
    while not(SelectedFeature) do
        Predicate = Generate next clause according to breadth first search;
        /*Link the predicate back to the target relation */ ;
        LinkedClause = Link(Predicate, B);
        /* Convert the LinkedClause into a feature that the statistical model can use */;
        NewFeature = aggregate(LinkedClause, T, S);
        F_new = F ∪ NewFeature;
        M_new = BuildTANNetwork(T, F_new);
        NewScore =AreaUnderPRCurve(M, S, F_new);
        /*Retain this feature*/ ;
        if NewScore > p * BestScore then
            F = F_new;
            BestScore = NewScore;
            M = M_new;
            Add predicate into background knowledge;
            SelectedFeature = true;
        end
    end
end
```

Algorithm 1: SAYU-VISTA

65365 negative examples. We used the same version of the data as Davis et al. [2005b]. We define *abnormality*, *visit* and *patient* as keys that can appear in the head of the clause. We aggregate over abnormalities.

5 Experiments and Results

We compare SAYU-VISTA to two SRL systems in our experiment. First, we compare SAYU-VISTA to SAYU [Davis *et al.*, 2005a] as it is the state-of-the-art view learning implementation, a follow-up to the original view learning paper [Davis *et al.*, 2005b]. However, SAYU only learns additional fields for existing tables; these fields are defined by learned rules that are approximations to the target concept. We also compare SAYU-VISTA against another leading SRL system—one that already has been applied with success (as measured by cross-validated precision-recall curves) to two of our application tasks and that has been receiving considerable attention: Markov Logic Networks [Richardson and Domingos, 2006], publicly available as the Alchemy system. Finally, we compared these three SRL systems against Aleph on all three data sets, and the SRL systems significantly outperformed Aleph on all three data sets. Therefore, to simplify the presentation we limit the remaining discussion to the three SRL systems.

All three SRL systems are evaluated by precision-recall curves estimated by cross-validation with significance of differences tested by a paired two-tailed t-test on areas under the precision-recall curves (AUCPR) across the different folds. We are careful to repeat any tuning of parameters on each fold of cross-validation, without looking at the test set for that fold, by dividing the data into a training set and tuning set. In this we follow the methodology of the developers of both MLNs and SAYU. For SAYU-VISTA, as for SAYU, we use the training set to learn the network parameters, while we use the tuning set to score potential clauses. For all datasets we use AUCPR as our score metric. However, we only look at AUCPR for recalls ≥ 0.5. We do this for two reasons. First, precision can have high variance at low levels of recall. Second, in domains such as Mammography, we are only interested in high levels of recall. A practicing radiologist would need to achieve at least this level of recall. A clause must improve the AUCPR (for recall ≥ 0.5) by at least 2% in order to be retained in the network. This is an arbitrary parameter setting; in fact we did not try any other thresholds. We had a time-based stop criteria for both SAYU and SAYU-VISTA. For UW-CSE each fold was given two hours to run, whereas for Mammography and Cora each fold received three hours runtime. We gave UW-CSE less time because it was a smaller data set. In practice, the time is not a limiting factor because few changes occur after the first 30 minutes for any of the tasks. MLN runs were not time-bounded. To offset potential differences in computer speeds, all experiments were run on identically configured machines.

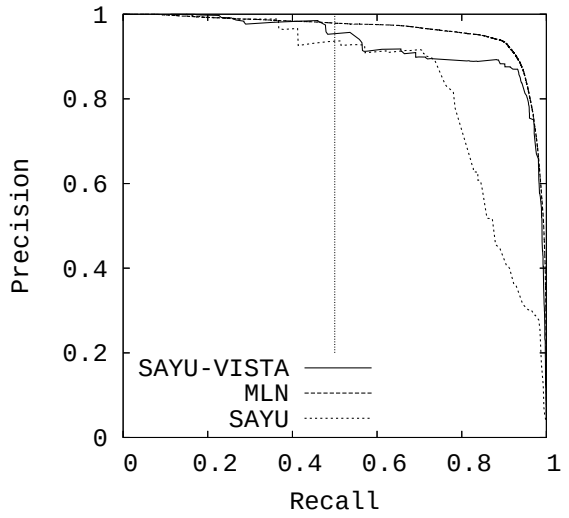

Figure 1: Precision-Recall Curves comparing SAYU-VISTA and SAYU on Cora

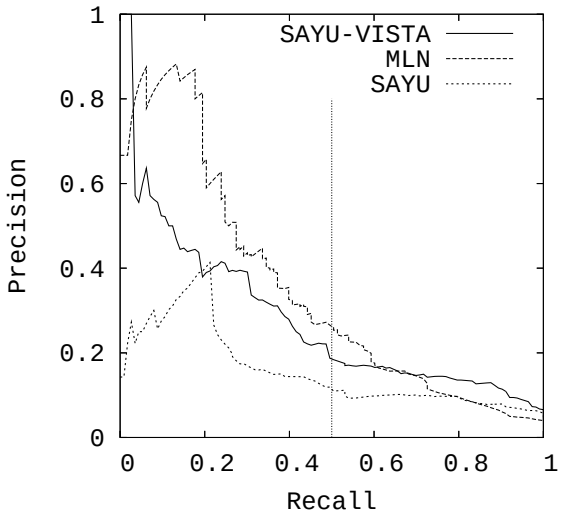

Figure 2: Precision-Recall Curves comparing SAYU-VISTA, MLNs, and SAYU on UW-CSE

We employed the default structure learning algorithm for MLNs and performed limited manual tuning of the parameters of the system to maximize AUCPR, while maintaining acceptable execution times. We report the best AUCPR values we obtained, over all attempted parameter settings. Note that we did not do any parameter tuning for SAYU-VISTA. The average, per-fold run-times for MLNs were all significantly longer than for either SAYU or SAYU-VISTA. The average, per fold run time for learning structure were five hours for Cora, seven hours for UW-CSE and three hours for Mammography.

5.1 Discussion of Results

Cora. Following Kok and Domingos [Kok and Domingos, 2005] we perform two-fold cross validation on this dataset for five different random train-test splits. We divide the training set in half, to form a new training set and a tuning set. Each fold received three hours of CPU time to run. SAYU and SAYU-VISTA can evaluate up to 300 clauses, before a selecting a new seed or clause head.

Table 1 reports the average AUCPR (recall ≥ 0.5) for Cora and the p-value for a two-tailed paired t-test between SAYU-VISTA and the other two algorithms. SAYU-VISTA performs significantly better than SAYU on this domain. Figure 1 shows precision-recall curves for all algorithms on this dataset. We pooled results across all folds to generate the curves. SAYU-VISTA dominates SAYU throughout precision-recall space. However, MLNs have a slightly higher average AUCPR than SAYU-VISTA does, although the difference is not significant. MLNs received an advantage over SAYU and SAYU-VISTA in this task, as MLNs started with an expert knowledge base.

UW-CSE. Following Richardson and Domingos [Richardson and Domingos, 2006], we perform five-fold cross validation on the UW-CSE dataset. We used two folds for the training set and two folds for a tuning set. Each approach could evaluate up to 10000 clauses, before either selecting a new seed (SAYU) or a new predicate head.

Table 1 reports the average AUCPR for UW-CSE and the p-value for a two-tailed paired t-test comparing SAYU-VISTA to the other approaches. SAYU-VISTA comes close to performing significantly ($0.05 < p < 0.06$) better than SAYU on this domain. Although performance varies widely between the 5 folds, SAYU-VISTA had a higher AUCPR than SAYU on each fold. SAYU-VISTA also comes close to outperforming MLNs on this data set, winning on four out of five folds.

Figure 2 shows precision-recall curves for SAYU, SAYU-VISTA and MLNs on this dataset. We pooled results across all five folds to generate the curves. Even though we measured AUCPR for recall ≥ 0.5, SAYU-VISTA dominates SAYU for most levels of recall. However, MLNs dominate SAYU-VISTA for low levels of recall, whereas SAYU-VISTA tends to dominate for the high levels of recall. We also compared the performance of SAYU-VISTA (average AUCPR of 0.468) and MLNs (average AUCPR of 0.355) for AUCPR for all levels of recall. Again, there is no significant difference. SAYU-VISTA has a higher variation in per fold AUCPR score than MLNs do. One reason for SAYU-VISTA's increased performance for high recall is that we are expressly optimizing for this metric. MLNs also receive one advantage over SAYU and SAYU-VISTA in this domain, in that they start with an expert defined knowledge base.

Mammography. Following Davis et al. [Davis *et al.*, 2005b] we perform ten-fold cross validation on this dataset. We used four folds for a training set and five folds as a tuning set. Each algorithm can evaluate at most 300 clauses for a given seed (SAYU) or clause head (SAYU-VISTA).

For the previous two datasets, we initially started with a Bayesian network that only contained a feature for the target predicate. However, in the Mammography domain we have access to a set of expert defined features (from the NMD). Furthermore, we could define a set of aggregate features as Davis et al. [Davis *et al.*, 2005b] did. Opposed to starting

	MLN	SAYU	SAYU-VISTA	p-value (vs SAYU)	p-value (vs. MLN)
Cora	0.4681	.3709	.4608	0.0109	0.3093
UW-CSE	0.06219	0.09747	0.1672	0.05807	0.1651
Mammography	0.0172	0.10337	0.1038	0.9693	$5.89*10^6$

Table 1: Average AUCPR for recall ≥ 0.5 for each task using TAN as the statistical model.

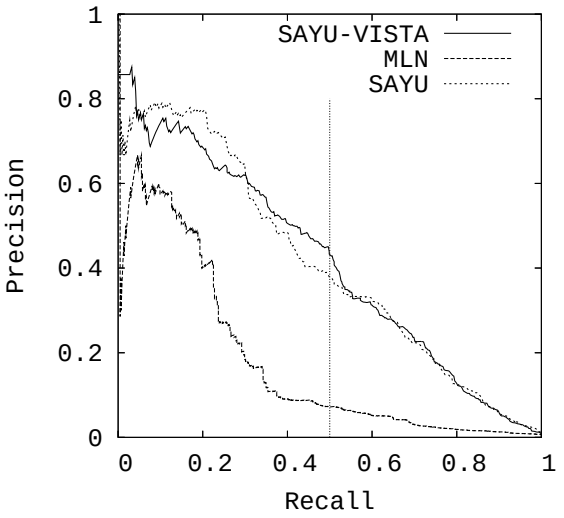

Figure 3: Precision-Recall Curves comparing SAYU-VISTA, MLNs and SAYU on Mammography

with an empty network structure, we begin with a network that contained both NMD features and the aggregate features.

Table 1 reports the average AUCPR over all folds. We use a two-tailed paired t-test to compute significant results, and the p-value for the test can also be found in Table 1. We find no significant difference between SAYU-VISTA and SAYU on this task, yet SAYU-VISTA does not perform any worse than SAYU. However, both SAYU and SAYU-VISTA significantly outperform MLNs on this domain.

Figure 3 shows precision-recall curves for both algorithms on this dataset. We pooled results across all folds to generate the curves. On this dataset, SAYU-VISTA and SAYU have comparable performance for all levels of recalls. SAYU-VISTA and SAYU both dominate MLNs for all levels of recall. Note that, as in the UW-CSE domain, MLNs tend to have better performance for low levels of recall. We feel there are several potential reasons that SAYU-VISTA did not perform significantly better than SAYU on this domain. First, for this application, MLNs and SAYU receive a large number of features—the precomputed aggregates—that SAYU-VISTA could potentially learn, but MLNs and SAYU cannot easily capture. Second, this domain contains many more constants than other domains, thus by leveraging Aleph, SAYU has a smaller and more directed search. Finally, the mammography domain contains on three relations, while while the other domains each have approximately twenty relations. Thus, on those domains there is more room to exploit the ability to learn predicates that (1) have different types in the head and (2) represent new tables.

In order to allow MLNs to run on this domain, we had to do drastic sub-sampling of the negative examples. MLNs struggled with having to ground out a network with the large number of examples that this data set contains. Another possible explanation for SAYU and SAYU-VISTA's better performance is that we seed the algorithm with an initial feature set. However, we ran the experiments where we started SAYU and SAYU-VISTA with an empty network structure and it still significantly outperformed MLNs.

5.2 Further Investigation of SAYU-VISTA

SAYU-VISTA adds several components to SAYU. First, it adds count aggregation: the ability to handle many-to-many and one-to-many relationships by adding a feature to the statistical model that counts the number of satisfying assignments for a predicate. Second, linkages allow us to learn entirely new tables. Third, we allow for previously invented predicates to appear in the definitions of new predicates. Without linkages, SAYU-VISTA reduces to SAYU. To discover the extent to which the other two features contribute to SAYU-VISTA's performance, we consider removing the first and third components from SAYU-VISTA and look at the resulting performance.

The first component, counting the number of satisfying assignments, does not help in either Cora or the Mammography domain. It is never used in Cora and it is only used twice in Mammography. Consequently, we do not need to consider removing it on these domains. However, it appears 11 times, or about twice per fold in the UW-CSE domain. Removing it reduces the AUCPR for this domain from 0.1520 to 0.1418. This degrades performance on four out of five folds, yet, the change is not significant, having a p-value of 0.16. However, it seems that even though counting does not help on two out of three domains, it can potentially be useful for an SRL system.

The other component of SAYU-VISTA we remove is the third, that of adding the learned predicates into background knowledge. Disabling this feature slightly improves performance in Mammography, increasing AUCPR from 0.1038 to 0.1046. However in Cora it decreasing AUCPR from 0.4608 to 0.448 and in UW-CSE the performance declines from 0.152 to 0.1475. Across all these experiments none of the changes are significant.

On Cora, the benefit comes only from the introduction of linkages. On UW-CSE, the benefit comes from both linkages and the count aggregation. In a sense linkages are the key innovation of SAYU-VISTA. Linkages allow us to both learn new tables and to learn concepts that are not simply approximations to the target concept. Reusing learned predicates

does not seem to provide a win. Asserting each learned predicate might unnecessarily widen the search space.

6 Related Work and Conclusions

We already have discussed how the present paper advances the state-of-the-art in view learning. The paper also is related to propositionalization within ILP [Lavrac *et al.*, 1991], particularly to propositionalization approaches that incorporate aggregation [Krogel and Wrobel, 2001; Knobbe *et al.*, 2001; Popescul *et al.*, 2003; Popescul and Ungar, 2004]. In these approaches, clause bodies are constructed that define new features or propositions. The value of such a feature for a data point, or example, is obtained by binding one of the variables in the clause body to the example's key, and then aggregating over the remaining features. In this fashion, the definition of a feature is equivalent to a definite clause whose head is "p(X)", where p is an arbitrary predicate name and X is the body variable that is bound in turn to each example's key. Both existential and count aggregation have been employed before [Krogel and Wrobel, 2001]. In fact, all the approaches cited above have used more complex aggregations than does SAYU-VISTA, and these could be incorporated easily into SAYU-VISTA. The novel properties of SAYU-VISTA relative to propositionalization by aggregation are the following. First, *subsets* of the variables in the clause body may be mapped back to an example's key, via the domain-specific linkage relations, thus enabling new tables or *non-unary* predicates to be learned, having different arities and types than the examples. Second, each time a potential new table or predicate is scored, an entire statistical model is constructed, and the new predicate is retained only if yields an improved model. Third, once a predicate is learned, it is available for use in the definitions of further new predicates. Although one piece of work cited above [Popescul and Ungar, 2004] does in fact allow some further use of some learned predicates, these new predicates are based on clustering and are constructed in an initial pre-processing step, before the learning of predicates to define new features for the statistical model; the latter features are never re-used.

Other general areas of related work are of course constructive induction [Rendell, 1985] and predicate invention [Muggleton and Buntine, 1988; Zelle *et al.*, 1994], as well as learning latent or hidden variables in Bayesian networks [Connolly, 1993]. Predicate invention is a specific type of constructive induction, where a new predicate is defined not based directly on examples of that predicate, but on the ability of that predicate to help in learning the definitions of other predicates for which examples are available. The classic difficulties with predicate invention are that, unless predicate invention is strongly constrained: (1) the search space of possible predicates is too large, (2) too many new predicates are retained, thus reducing efficiency of learning, and (3) the ability to invent arbitrary new predicates leads to overfitting of training data.

The present work can be seen as a type of predicate invention, because arbitrary clauses are constructed whose heads do not have to unify with the examples—they may have arities and types different from the examples. SAYU-VISTA is analogous to CHILLIN [Zelle *et al.*, 1994] and Closed World Specialisation [Srinivasan *et al.*, 1992]. Both of those systems search for an intensional definition of a clause based on existing predicates, just like SAYU-VISTA. One key difference is that those systems don't directly search for new predicates. CHILLIN is demand driven, and Closed World Specialisation invents predicates to handle exceptions to the theory, whereas SAYU-VISTA directly searches for new predicates. The other important difference is how the systems evaluate new predicates. The other systems use traditional ILP metrics, such as compaction. The approach in the present paper is to constrain predicate invention by requiring invented predicates to be of immediate value to the statistical learner in order to be retained for further use. The empirical success of SAYU-VISTA—that it does not hurt performance and it sometimes helps—indicates that this efficacy test is a successful constraint on predicate invention.

The topic of learning Bayesian network structures with the introduction of new (latent) variables faces similar obstacles to predicate invention. Because the new variables are unconstrained by the data, their introduction into Bayesian network structure learning permits overfitting of the training data, in addition to increasing search complexity. SAYU-VISTA may be seen as introducing new variables into the structure learning task; nevertheless, by requiring these new variables to be defined using existing (pre-defined or recently learned) relations, these variables are partially constrained. The empirical success of SAYU-VISTA provides some evidence that this constraint on the new variables helps to avoid overfitting. SAYU-VISTA's use of TAN Bayes nets also helps to reduce the search space. Both predicate invention and Bayes net learning with the introduction of new variables are widely noted to be extremely difficult tasks. This paper provides some evidence that attempting to address both tasks at the same time, within an SRL framework, can actually make both tasks somewhat easier.

Acknowledgements

This work was supported in part by U.S. National Science Foundation grant IIS 0534908 and by an NLM training grant to the Computation and Informatics in Biology and Medicine Training Program (NLM 5T15LM007359). Jan Struyf is a postdoctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). We would also like to thank Pedro Domingos, Stanley Kok and the rest of Alchemy team for answering our questions regarding the Alchemy system and for providing the Cora dataset.

References

[ACR, 2004] ACR. Breast imaging reporting and data system (birads), 2004.

[Connolly, 1993] D. Connolly. Constructing hidden variables in bayesian networks via conceptual clustering. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 65–72, 1993.

[Davis *et al.*, 2005a] J. Davis, E. Burnside, I.C. Dutra, D. Page, and V. Santos Costa. An integrated approach to learning bayesian networks of rules. In *16th ECML*, pages 84–95. Springer, 2005.

[Davis *et al.*, 2005b] J. Davis, E. Burnside, I.C. Dutra, D. Page, R. Ramakrishnan, V. Santos Costa, and J. Shavlik. View learning for statistical relational learning: With an application to mammography. In *Proceedings of the 19th IJCAI*, pages 677–683, Edinburgh, Scotland, 2005.

[Friedman *et al.*, 1997] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian networks classifiers. *Machine Learning*, 29:131–163, 1997.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th IJCAI*. Stockholm, Sweden, 1999.

[Knobbe *et al.*, 2001] A. Knobbe, M. de Haas, and A. Siebes. Propositionalisation and aggregates. In *Proceeding of the 5th PKDD*, pages 277–288, 2001.

[Kok and Domingos, 2005] S. Kok and P. Domingos. Learning the structure of markov logic networks. In *Proceedings of the 22nd ICML*, pages 441–448. ACM Press, 2005.

[Krogel and Wrobel, 2001] M. Krogel and S. Wrobel. Transformation-based learning using multirelational aggregation. In *Proceedings of the 11th ILP*, pages 142–155, London, UK, 2001. Springer-Verlag.

[Landwehr *et al.*, 2005] N. Landwehr, K. Kersting, and L. De Raedt. nFOIL: Integrating Naive Bayes and FOIL. In *Proceeding of AAAI*, 2005.

[Lavrac *et al.*, 1991] N. Lavrac, S. Dzeroski, and M. Grobelnik. Learning non-recursive definitions of relations with LINUS. In Y. Kodratoff, editor, *Proceedings of the Fifth European Working Session on Learning, LNAI 482*, pages 265–281. Springer-Verlag, 1991.

[McCallum *et al.*, 2000] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000. www.research.whizbang.com/data.

[Muggleton and Buntine, 1988] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352. Kaufmann, 1988.

[Muggleton, 1995] S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[Popescul and Ungar, 2004] A. Popescul and L. Ungar. Cluster-based concept invention for statistical relational learning. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the 10th SIGKDD*, pages 665–670, 2004.

[Popescul *et al.*, 2003] A. Popescul, L. Ungar, S. Lawrence, and D. Pennock. Statistical relational learning for document mining. In *Proceeding of the 3rd ICDM*, pages 275–282, 2003.

[Rendell, 1985] L. Rendell. Substantial constructive induction using layered information compression: tractable feature formation in search. In *IJCAI-85*, pages 650–658. Kaufmann, 1985.

[Richardson and Domingos, 2006] M. Richardson and P. Domingos. Markov logic networks. *To appear in Machine Learning*, 2006.

[Srinivasan *et al.*, 1992] A. Srinivasan, S. Muggleton, and M. Bain. Distinguishing exeptions from noise in non-monotonic learning. In *Proceedings of the 2nd ILP workshop*, 1992.

[Srinivasan, 2001] A. Srinivasan. *The Aleph Manual*, 2001.

[Zelle *et al.*, 1994] J. Zelle, R. Mooney, and J. Konvisser. Combining top-down and bottom-up techniques in inductive logic programming. In *Proceedings of the 11th ICMl*, pages 343–351, 1994.