Comparative Evaluation of PL Languages and Systems – a Work-in-Progress Report

Manfred Jaeger[1], Petr Lidman[2], and Juan L. Mateo[3]

[1] Institut for Datalogi, Aalborg Universitet, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø
jaeger@cs.aau.dk

[2] Faculty of Informatics, Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
lidman@fi.muni.cz

[3] Computing Systems Department and SIMD-$i^3\mathcal{A}$, University of Castilla-La Mancha, 02071, Albacete, Spain
juanlmc@dsi.uclm.es

Abstract. We introduce a framework for an empirical evaluation of probabilistic logic languages and systems.

1 Introduction

The field of probabilistic logic languages for knowledge representation and learning (also known as *statistical relational learning*) is developing fast, and an increasing number of languages supported by prototype implementations is being proposed. In order to obtain a better understanding of the common nature of these languages, as well as of salient differences that can make one language preferable over another for a specific application, it is desirable to establish a platform on which languages can be compared with regard to their expressivity, complexity, and learnability. First steps towards a theoretical framework for such a comparative analysis have already been taken [1, 4]. With this paper we introduce a platform for a practical, experimental comparison of systems that implement a variety of different pl-languages. It is important to emphasize that even though we would like to compare languages (or better still, basic representation paradigms that are more or less purely represented by some concrete pl-language), we only can compare systems. The implemented systems often represent a mixture of the pure underlying language design and pragmatic implementation solutions. The goal of this comparison is to evaluate the suitability and performance of different systems on a suite of challenge problems. The first step towards this goal is to identify a general form of 'problems' for pl-systems (Section 2).

We then formulate four concrete challenge problems (Section 3), and investigate how four different pl-systems handle these problems: the Alchemy system[4], implementing *Markov Logic Networks (MLNs)* [6]; the Balios system [5], implementing *Bayesian Logic Programs (BLPs)* [5], the Primula system [6], implementing *Relational Bayesian Networks (RBNs)* [2], and the Prism system[7] [7]). The selection of these systems for an initial study is based on the relatively large overlap of their representation and inference capabilities. In further investigations, also somewhat less closely related systems like IBAL[8] and BLOG[9] should be included in the comparison.

In this initial report we are not able to give detailed, quantitative results on the performance of the different systems. We only give some general findings on the main issues and problems encountered so far. Main purpose of this paper is to introduce our platform for comparison, and to invite contributions from expert users and developers of the four systems considered here, as well as any other pl-system. To this end we have established a web-site (www.cs.aau.dk/∼jaeger/plsystems) where challenge problems can be posed, and solution models can be presented.

[4] alchemy.cs.washington.edu/
[5] www.informatik.uni-freiburg.de/∼kersting/profile/
[6] www.cs.aau.dk/∼jaeger/Primula/
[7] sato-www.cs.titech.ac.jp/prism/
[8] www.eecs.harvard.edu/ avi/IBAL/
[9] people.csail.mit.edu/milch/blog/

2 Establishing a Common Framework

A key prerequisite for a comparison of pl-languages is a common semantic framework that allows us to decide whether two models represented in different languages are equivalent. We use the framework introduced in [4] as the basis for our comparison. Central to this framework is the idea that a pl-model is composed of an *intensional*, and an *extensional* part. The intensional part contains the generic, high-level probabilistic model (e.g. general rules for the propagation of genetic traits), whereas the extensional part contains the specification of a concrete domain to which the model is applied (e.g. a specific set of individuals with their family relationships). A general, abstract model for such concrete domains are *relational structures*: a set of entities on which certain relations (possibly multi-valued) are defined. The pl-model then defines random attributes and relations on the set of domain entities (e.g. the random attribute “bloodtype” on the individuals of the input domain), i.e. it defines a probability distribution over relational structures again.

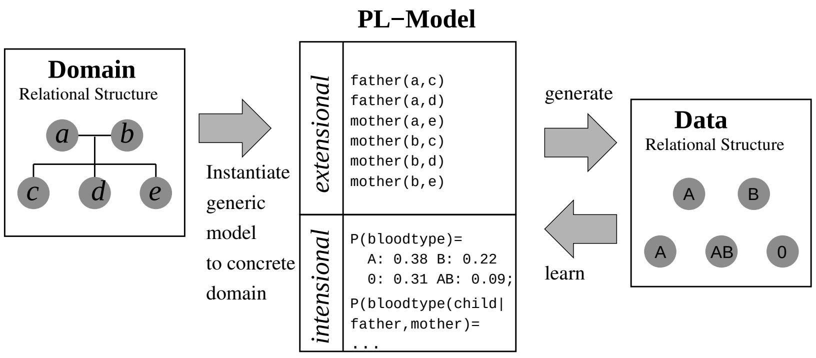

Fig. 1. PL models

The separation into intensional and extensional part is more or less clearly reflected in the syntax of the different languages we investigate. It is made most clearly in the RBN language. BLPs, too, make an explicit distinction between (logical) input relations, and probabilistic (output) relations. For Prism and MLNs there is no explicit syntactic separation, and it may therefore not always be possible to divide any given Prism or MLN model into an intensional and extensional part. However, conversely, given a pl-model which is language-independently defined in terms of intensional and extensional components, one will always find encodings of this model in Prism or MLN that in a modular way separately represent the intensional and extensional parts. In particular, for Prism and MLNs, too, it is easy to replace the specification of the input domain, without changing the underlying probabilistic model.

Finding equivalent models is the first step in a comparison of systems. A challenge problem also consists of the specification of inference and learning tasks that the models should support.

When we investigate complexity of probabilistic inference in the different systems, we focus mainly on scalability in terms of the size of the input domain. In order to compare efficiency, we have to encode equivalent models in the different languages, which, in particular, means to encode the same (potentially large) input domains as extensional parts in different languages. In order to facillitate the translation of these extensional parts, we have defined a general XML format for the representation of relational structures, and implemented a translator that takes an XML specification of an input domain, and translates it into an extensional model part in any of the four target languages. Since data, too, is given by relational structures, the same XML format is also used to represent data for learning, and our translator can also transform an XML specification into the data format required by the different systems.

3 Four Challenge Problems

To compare the systems we select a number of “challenge problems”. Each challenge problem consists of a model that has been given as an example for one of the systems. The original example model serves as the reference model, and the goal is to find an equivalent representation in other languages. In all cases we require models that support the inference task of computing posterior marginals for unobserved variables, given instantiations of arbitrary observed variables. The following challenge problems are selected to be rather simple examples for fundamental modeling tasks. The scope of challenge problems to be considered will eventually be enlarged to also include e.g. probabilistic context-free grammars.

Bloodtype model: A simple genetic model for the inheritance of bloodtypes in a pedigree. Input domains consist of pedigrees with individuals and a “father” and “mother” relation. The reference model comes from Balios.

University model: A model for the adviser relation between students and professors. Input domains consist of a number of students and professors. This model is representative for undirected models given by feature-induced potentials, rather than conditional probabilities. The reference model comes from Alchemy.

HMM model: A standard HMM model. Input domains just consist of the number of time slices over which the model is to be developed (this need not necessarily be explicitly specified as an extensional model part). We use several variants of an HMM model that differ in the number of states of the hidden variable, and investigate how this affects the representability of this model in languages that only permit binary variables (Primula and Alchemy).

Noisy-or model: A very basic model that uses the noisy-or function for combining several causal inputs. Input structures are directed acyclic graphs on which a random unary attribute is propagated from the roots downward. Reference model comes from Primula.

We have experimented with representations of these four models in the four systems under investigation. The results we obtained are too preliminary and inconclusive to already permit a detailed, quantitative comparison of the systems. Moreover, it has become quite clear that such a comparison will require input from expert users and developers of the various systems, because the performance of an encoding of a challenge problem can vary substantially according to the sophistication of the encoding. However, our experience so far allows us to identify a number of points that seem to be characteristic for the different systems. First, it proved rather difficult to represent some of our challenge problems in Alchemy. To some extent this is to be expected, as all except the University model are directed probabilistic models, whereas Alchemy most naturally represents undirected models. This should not be a fundamental problem, however, as it is known how to represent directed models as MLNs [6]. The main reason for the difficulties is the fact that Alchemy does not implement the full MLN language, but only its “clausal fragment”, and direct representations of all of our models require non-clausal first-order formulas. Prism, too, is a system which is not ideally tailored for the type of problems we considered. The main problem for Prism was that the system does not have the built-in functionality to support the type of queries we were interested in, i.e., computation of posterior marginals given arbitrary instantiations of some variables as evidence. Support for such queries must be explicitly programmed into the Prism model. Generally, Prism more than the other systems, is a programming language that requires a procedural model representation, whereas Alchemy, Balios and Primula all are essentially declarative knowledge representation languages. However, efficient model representation in these languages too, requires some knowledge of the procedural aspects of the system’s inference procedures.

When evaluating inference efficiency and accuracy, we found that both in Balios and in Alchemy the obtained inference results sometimes depended strongly on the option settings for inference. For example, Gibbs and rejection sampling (in Balios), or Gibbs sampling and MC-SAT (in Alchemy) sometimes gave widely differing results, possibly due to insufficient mixing of the Markov chains.

4 Binarization

One issue we investigated in greater depth was the problem of representing multi-valued relations in a language that only allows Boolean relations (RBNs, MLNs). We studied this problem using variations of the HMM model with hidden state variables ranging from 8 to 64 states. It is most commonly suggested to binarize a multi-valued relation like *hidden*$(t) = s_i$ $(i = 1, \ldots, n)$ by turning the relation value into an additional argument (*hidden*$(t, s_i) =$ *true*) [3,6]. However, this approach requires to also encode in the model the constraint that for each argument t the relation *hidden*(t, s_i) is true for exactly one s_i. This induces a mutual dependency of the variables *hidden*(t, s_i), which will typically lead to an exponential behavior in inference.

A more efficient way of binarization of multi-valued relations is to identify the set of n possible values with bitstrings of length $\lceil log_2 n \rceil$, and to introduce for each bit in the bitstring one boolean predicate. Assuming, for example, that $n = 64$, we introduce predicates *hidden_bit1*$(t), \ldots,$ *hidden_bit6*(t). Each instantiation of these boolean predicates identifies one possible state (when n is not a power of 2, one may take some bitstrings to represent dummy states of zero probability). Using these bitstring predicates, we have implemented the HMM model with $n = 8, 16, 32, 64$ in Primula, and found that (exact) inference exhibited behavior consistent with $O(n^2)$ complexity, which is to be expected, as we are essentially dealing with transition matrices of size n^2.

While computationally feasible, the boolean bitstring representation is somewhat hard to code manually. In order to make general use of it in the context of more complex models than HMM, one would need to automate the compilation process of multi-valued predicate representations into purely boolean representations.

5 Conclusion

We have established a platform for comparison and benchmarking of pl-systems, and reported some very preliminary findings on the performance of four different systems on four different challenge problems. With this paper we hope to stimulate the participation of other researchers in this empirical evaluation. To facillitate the exchange of problems and solutions, we have created an XML file format for the exchange of relational structures and relational data, and implemented a utility program for translating specifications in this format into the formats used by various systems. These resources, exact challenge problem specifications, and solution models are provided on a web-site (www.cs.aau.dk/∼jaeger/plsystems), which is meant to become an open platform for the exchange of challenge problems and solutions.

References

1. J. Chen and S.H. Muggleton. A revised comparison of Bayesian logic programs and stochastic logic programs. In *Short Paper Proceedings of the 16th International Conference on Inductive Logic Programming*, 2006.
2. M. Jaeger. Relational bayesian networks. In Dan Geiger and Prakash Pundalik Shenoy, editors, *Proceedings of the 13th Conference of Uncertainty in Artificial Intelligence (UAI-13)*, pages 266–273, Providence, USA, 1997. Morgan Kaufmann.
3. M. Jaeger. Complex probabilistic modeling with recursive relational Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32:179–220, 2001.
4. M. Jaeger, K. Kersting, and L. De Raedt. Expressivity analysis for pl-languages (position paper). In *Online Proceedings of the Workshop on Statistical Relational Learning (SRL-06)*, 2006.
5. K. Kersting and L. De Raedt. Towards combining inductive logic programming and bayesian networks. In *Proceedings of the Eleventh International Conference on Inductive Logic Programming (ILP-2001)*, Springer Lecture Notes in AI 2157, 2001.
6. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107 – 136, 2006.
7. T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.