Multivariate Prediction for Learning in Relational Graphs

Yi Huang and Volker Tresp
Siemens AG, Corporate Technology
Otto-Hahn-Ring 6, 81739 München, Germany
YiHuang{Volker.Tresp}@siemens.com

Hans-Peter Kriegel
Institute for Computer Science
Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München, Germany
kriegel@dbs.ifi.lmu.de

Abstract

This paper concerns learning with data in relational formats. In focus are simplicity, scalability and ease of use. Inspired by the graphical representation used in context of the RDF data model of the Semantic Web, we propose a graphical representation for the data in relational data bases. Based on this relational graphical model, we define a statistical learning framework and derive a data matrix on which machine learning is applicable. The data matrix typically contains a large number of coupled random variables (outputs) and a large number of covariates (inputs). We discuss learning algorithms from multivariate prediction, which we utilize to estimate the truth values of tuples not yet present in the data base. Learned tuples and their certainty values can be stored in the data base and can be integrated in querying. We present experimental results using data from a social network side.

1 Introduction

Relational learning is an area of increasing interest mainly due to the growing availability of linked data, e.g., in the form of hyperlinked World Wide Web documents, of citation data, of social network data and of various networks in the life sciences. More specifically, the term Linked Data is used in context of the Semantic Web (SW)[1] as the effort to link and jointly explore several structured data sources on the Web[2]. A prominent example is the LinkedLifeData project[3], in which a subset of the more than 1000 publicly available life science data bases can jointly be queried. Linked Data become increasingly interesting as an application area for machine learning. As the name implies, to a great extent Linked Data describe relations between objects and thus relational learning should immediately be applicable. Relational learning must face the challenges raised by the typical data situation of Linked Data. First, data are heterogeneous with many different entity types and relationships, often arranged in a hierarchical ontology. Second, relations are often extremely sparse, e.g., only a tiny subset of all possible persons are someone's friends. Third, e.g., for privacy reasons, information is missing. In addition we require machine learning to be highly scalable, which excludes many theoretically interesting relational learning approaches. We also want machine learning to be easily configurable since it is often applied in an explorative matter and utilized by people who are not experts in machine learning. Within the EU FP7 project LarKC [9], a machine learning approach called SUNS (Statistical Unit Node Set) is being developed that attempts to conquer all those challenges by learning on the SW [17]. In this paper we extend the SUNS approach to be applicable in the context of relational data bases, mainly for two reasons. First, although the SW might provide

[1]http://www.w3.org/2001/sw/
[2]http://linkeddata.org/
[3]http://www.linkedlifedata.com/

the data models of the future, currently the majority of data resides in relational data bases. Second, the RDF (Resource Description Framework)[4] directly represents only binary relations; to represent higher order relations, awkward, so called blank nodes need to be introduced. No such restriction exists for data in a relational format.

A great advantage of the RDF data model is the simple semantics transported by its graphical representation: a complete data base can be represented as a directed graph where nodes stand for resources (e.g., objects) and links denote relations between resources. In this paper we introduce a similarly simple and intuitive graphical representation for a relational data base (RDB) containing relations with arbitrary arity. We assume the data base has been transformed into the fifth normal form (5NF). We map the scalable SUNS approach to this representation by defining a statistical setting, in which statistical inference is well defined. We derive a data matrix to which multivariate modeling is applicable and estimate the truth values of tuples not yet present in the data base. Learned tuples and their certainty values can be stored in the data base and can then be integrated in querying.

The paper is organized as follows. In the next section, we discuss related work. In Section 3 we introduce our graphical representation of a relational data base. In Section 4 we define the statistical setting, define the random variables in the domain of interest, and derive the data matrix. In Section 5 we present experimental results. Our conclusions are presented in Section 6.

2 Related Work

The approach is an extension of the work presented in [17] for learning with the RDF data model. Here we use a relational representation, which is able to encode relations with arity larger than two. Inductive logic programming (ILP) has traditionally been applied to relational domains [15, 13]. The difference is that classical ILP is concerned with deterministic or close to deterministic dependencies whereas here we are concerned with statistical learning. Also, ILP requires a complex feature selection step, which we avoid. Propositionalization [3, 10] is an ILP approach, which is closer to our view. We see advantages in our approach due to its simplicity, generality, ease of use and scalability. Also in contrast to typical approaches in propositionalization, we apply multivariate prediction, which allows us to predict jointly a large number of tuples in one step. Multivariate prediction has been shown to give superior performance in predicting relationships between objects (e.g., between users and movies). Learning frameworks in relational data bases are also being pursued in industry. For example, Microsoft's Data Mining Extension (DMX) defines a general approach for learning in relational data bases. A difference is that we are mostly interested in predicting relationships between objects by using multivariate models, whereas most common frameworks focus on standard data mining tasks such as clustering and classification. The work in SRL (Statistical Relation Learning) [5, 4, 19, 7, 14] is certainly also closely related and the main advantage of our approach is that we do not require an involved structural search procedure, as, for example, PRMs (probabilistoc relational models) [5]. In contrast to MLNs (Markov Logic Networks) [16], we do not need as a basis rules or other logical constraints defined by a domain expert.

3 A Graphical Representation of a Relational Data Base

The data format of the SW is the RDF-graph, in which resources (i.e., objects) are linked by directed arcs describing simple subject-predicate-object statements. Since the graphical representation in form of the RDF-graph has proven to be very useful for the SW, we introduce now a graphical representation of a relational data base (RDB), i.e., a relational graph. The graphical representation is convenient for the derivation of the data matrix but, naturally, the approach is independent of this graphical representation. Whereas an RDF-graph directly presents only binary relationships, the relational graph can represent relations with arbitrary arity. An advantage of our representation is that random variables are explicitly represented as nodes, which is neither true in the RDF-graph nor in many other relational graphical representations.

[4]http://www.w3.org/RDF/

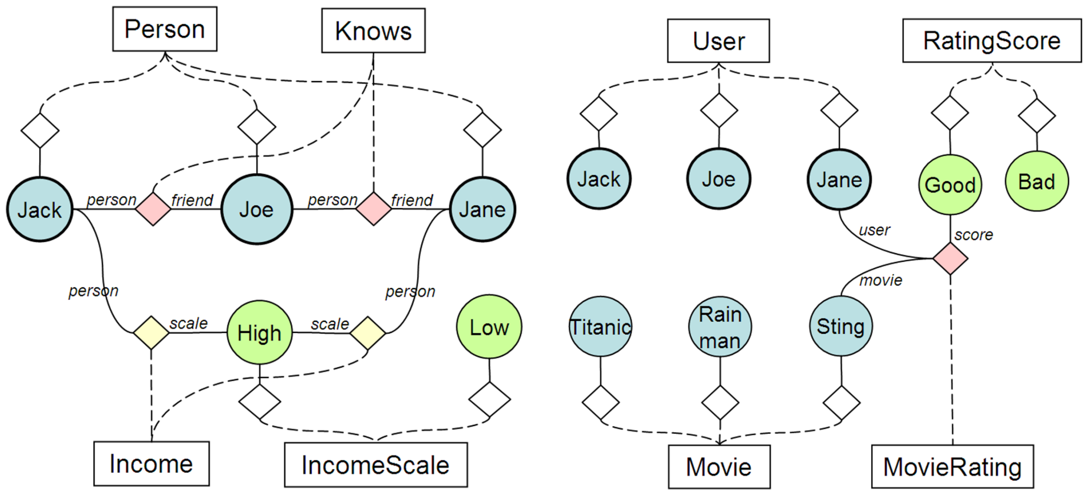

Figure 1: Relational Graphs. Constants are represented by circular nodes, tuples by diamond-shaped nodes and relations by rectangular nodes. Nodes representing statistical units (in both cases: *Persons*) have a darker rim. The dashed lines indicate which tuple belongs to which relation. Constants are linked to the tuples they appear in and the links are labeled by the attribute names. The left graph shows a social friendship network in which the income of a person is an attribute. The right side shows a movie rating domain.

For simplicity of exposure we make the following assumptions, which can partially be relaxed. First, we assume that each object has a UID, i.e., a unique identifier. Secondly, we assume that the underlying RDB is minimal in the sense that the arity of a relation cannot be reduced without changing its semantics. An example: the relation *MovieRating(Jane, Sting, Good)* cannot be reduced. In contrast, *PersonAttributes(Jane, Blond, Tall)* can be reduced to *HairColor(Jane, Blond)* and *PersonHight(Jane, Tall)* without loss of information. The reason is that higher order relationships are more difficult to handle for machine learning and should be avoided if possible. In the theory of data base normalization, this corresponds to the fifth normal form (5NF) [8]. Finally we make a distinction between constants that have object character (resources in SW terms, e.g., persons) and constants that have attribute character (literals in SW terms, e.g, numerical values, Good, Bad, etc.).

We illustrate the graphical relational graph using as an example a social network in Figure 1 (left). The constants in the data base are represented as circular nodes with object nodes *Jack*, *Joe* and *Jane* (blue) and attribute nodes *Low* and *High* (green). Next we introduce a rectangular node for each relation *Person, Knows, IncomeScale,* and *Income*. For each tuple in the RDB a diamond-shaped node is introduced. A tuple node is connected via a dashed line to the associated relation and is linked to the constants in the tuple via solid lines. We draw unary tuples in white, tuples between object constants in purple and tuples between object constants and attribute constants in yellow. Finally, the links between the tuples and the objects are labeled by the corresponding attribute (column) name of the relation. The tuple nodes for unary relations, i.e., *Person, IncomeScale* are somewhat redundant but they are useful later when we introduce random variables. Given objects, relations and attributes, the random quantities are the tuples; thus we consider that tuples, which are not known to be true, exist with some probability.

4 Statistical Modeling

4.1 Defining the Sample

In a relational domain the issue of a probability distribution is somewhat tricky. Consider a set of physicians and patients, let Q be the quality of a physician, and assume that 50% of the physicians are good and 50% of the physicians are bad. If we sample physicians, we might conclude that $P(Q = Good) = P(Q = Bad) = 1/2$. On the other side patients might have a preference to visit good physicians. Thus, if we sample patients, we might obtain that $P(Q = Good) = 0.7$, i.e., that 70% of the physicians of the patients are good. Thereby one can conclude that in relational domains (as in any other domains), the interpretation of a probability distribution is determined by the sampling process. More precisely, we must be careful in defining the statistical unit, the population, the sampling procedure and the features. A statistical unit is an object of a certain type, e.g., a person. In our framework, a statistical unit might be an object in a unary relation, e.g., in the relation *Person*. The population is the set of statistical units under consideration. In our framework, a population might be defined as the set of persons that attend a particular university. For learning we use a subset of the population, typically by random sampling. Based on the sample, a data matrix is generated where the statistical units in the sample define the rows.

4.2 The Random Variables in the Data Matrix

We now introduce the state of a tuple node. A tuple node is in state *one* (*true*) it the tuple is known to exist and is in state *zero* (*false*) if the tuple is known not to exist. Graphically, one only draws the tuple nodes in state one, i.e., the existing tuples in the RDB. In RDBs one typically makes a closed-world assumption such that the tuples that are not represented in the RDB are assumed false. Here we assume that the truth values of the remaining tuples are unknown. This is in accordance with the view, for example, in MLNs [16].

We now associate some tuples with statistical units. In connection with the random sampling process described earlier tuple nodes become random variables with states *one* and *zero*. The idea is now to assign a tuple to a statistical unit if the statistical unit appears in the tuple. Let's consider the statistical unit *Jane* (Figure 1, left). Based on the tuples she is participating in we obtain the expressions *Person(X), Knows(Joe, X), Income(X, High)*, where X is a variable that represents a statistical unit. The expressions form the random variables (outputs) and define columns in the data matrix. By considering the remaining statistical units *Jack* and *Joe* generate the expressions (columns) *Knows(X, Jane), Knows(X, Joe), Knows(Jack, X)*. We will not add *Knows(Jane, X)* since Jane considers no one in the data base to be her friend. We iterate this procedure for all statistical units in the sample and add new expressions (i.e., columns in the data matrix), if necessary. Note, that expressions that are not represented in the sample will not be considered. Also, expressions that are rarely true (i.e., for few statistical units) will be removed since no meaningful statistics can be derived from a small sample.

An example for a ternary relation is shown in Figure 1 (right). Here, based on *Jane*'s tuples the expression *MovieRating(X, Sting, Good)* is added; it represents statistical units that like the movie *The Sting*.

In [17] the tuples associated with a statistical unit were denoted as statistical unit node set or SUNS.

4.3 Non-random Covariates in the Data Matrix

The columns we have derived so far represent tuples that belong to the schema. Those tuples are treated as random variables in the analysis. If machine learning predicts that a tuple is very likely, we can enter this tuple in the RDB (see Section 6). We now add columns that provide additional information for machine learning but which we treat as covariates or fixed inputs.

First, we derive simplified relations from the RDB. More precisely, we consider the expressions derived in the last subsection and replace constants by variables. For example, from *Knows(X, Jane)* we derive *Knows(X, Y)* and count how often this expression is true for a statistical unit X, i.e. we count the number of friends of a statistical unit X. From the ternary tuple *MovieRating(X, Sting, Good)* we obtain *MovieRating(X, Y, Good)*, *MovieRating(X, Sting, Y)*, and *MovieRating(X, Y, Z)*. By counting the occurrences we obtain the number of movies a statistical unit has rated as *Good*, how often a statistical unit has rated *The Sting*, and how often a statistical unit has rated any movie.

Second, we consider two simple types of aggregated features from outside a SUNS. Consider first a binary tuple *Knows(X, Jane)*. If Jane is part of another binary tuple, in the example, *Income(Jane, High)*, then we form the expression *Knows(X, Y)* $\wedge$ *Income(Y, High)* and count how many rich friends a statistical unit has. Consider a ternary tuple *MovieRating(X, Sting, Good)* with two objects and one attribute constant. Consider an additional binary tuple of the form *Starring(Sting, PaulNewman)*. Then we form *MovieRating(X, Y, Good)* $\wedge$ *LeadingActor(Y, PaulNewman)* and count how many

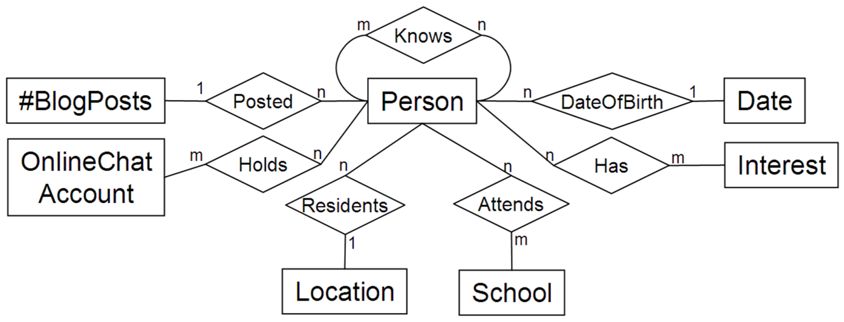

Figure 2: Entity-relationship diagram of the LJ-FOAF domain

Paul Newman movies a statistical unit X likes. A large number of additional aggregated features are possible but so far we restricted ourselves to these two types.

After construction of the data matrix we prune away columns which have ones in fewer than ϵ percent of all rows or in more than $(1-\epsilon)$ of all rows, where ϵ is usually a very small number. Thus we remove aggregates features that are very rarely true or almost always true, since for those no meaningful statistical analysis is possible. Note that by this pruning procedure we have reduced the exponential number of random variables to typically a much smaller set.

4.4 Algorithms for Learning with Statistical Units Node Sets

A row in the resulting data matrix contains external inputs based on aggregated information (if available) and typically a large number of binary and sparse outputs. A *one* stands for a tuple known to be true and a *zero* for a tuple whose truth value is unknown. In this situation, multivariate structured prediction approaches have been most successful [18]. In multivariate structured prediction all outputs are jointly predicted such that statistical strength can be shared between outputs. The reason is that some or all model parameters are sensitive to all outputs, improving the estimates of those parameters. The approaches we are employing here are based on a matrix completion of the complete data matrix, including inputs and outputs.[5] We investigate matrix completion based on a singular value decomposition (SVD), e.g., [12], matrix completion based on non-negative matrix factorization (NNMF) [11] and matrix completion using latent Dirichlet allocation (LDA) [1]. All three approaches estimate unknown matrix entries via a low-rank matrix approximation. SVD is based on a singular value decomposition and NNMF is a decomposition under the constraints that all terms in the factoring matrices are non-negative. LDA is based on a Bayesian treatment of a generative topic model. After matrix completion of the $zero$ entries in the data matrix, the entries are interpreted as certainty values that the corresponding tuples are true. After training, the models can be applied to statistical units in the population outside the sample.

5 Experiments

5.1 Data Set and Experimental Setup

The experiments are based on a friend-of-a-friend (FOAF) data set. The purpose of the FOAF project [2] is to create a web of machine-readable pages describing people, their relationships, and people's activities and interests.

We gathered our FOAF data set from user profiles of the community website LiveJournal.com[6]. The data set will be made available online. All extracted entities and relations are shown in Figure 2. In total we collected 32,062 persons and all related attributes. We selected 14,425 persons with "dense" friendship relationships. On average, a given person has 27 friends.

[5] Although the completion is applied to the complete matrix, only $zeros$ —representing tuples with unknown truth values— are overwritten.

[6] http://www.livejournal.com/bots/

Table 1: Best *NDCG all*: knows relation with threshold 10

		Threshold of attr. / aggr. attr.			
Setting	Method	0/0	10/50	20/200	inf/inf
inductive	SVD	0.3460 ± 0.0044	0.3211 ± 0.0040	0.3330 ± 0.0038	0.3155 ± 0.0041
	LDA	0.2804 ± 0.0022	0.2807 ± 0.0024	0.2967 ± 0.0023	0.3137 ± 0.0020
transductive	SVD	0.3446 ± 0.0092	0.3297 ± 0.0084	0.3349 ± 0.0094	0.3201 ± 0.0085
	LDA	0.3312 ± 0.0079	0.3265 ± 0.0078	0.3289 ± 0.0072	0.3393 ± 0.0073

Table 2: Best *NDCG all*: knows relation with threshold 20

		Threshold of attr. / aggr. attr.			
Setting	Method	0/0	10/50	20/200	inf/inf
inductive	SVD	0.4811 ± 0.0060	0.4657 ± 0.0058	0.4557 ± 0.0059	0.4424 ± 0.0061
	LDA	0.3809 ± 0.0047	0.4025 ± 0.0046	0.3742 ± 0.0046	0.4332 ± 0.0044
transductive	SVD	0.4703 ± 0.0073	0.4460 ± 0.0066	0.4453 ± 0.0065	0.4201 ± 0.0070
	LDA	0.4801 ± 0.0055	0.4790 ± 0.0056	0.4465 ± 0.0053	0.4705 ± 0.0052

The task is to predict potential friends of a person, i.e., $Knows$ tuples. For each person in the data set, we randomly selected one $Knows$ friendship tuple and set the corresponding matrix entry to $zero$, to be treated as unknown (test tuple). Also we consider the modification in the aggregated features. In the test phase we then predicted all unknown friendship entries, including the entry for the test tuple. The test tuple should obtain a high likelihood value, if compared to the other unknown friendship entries.

Aggregated features here describe the aggregated properties of persons to which a *Knows* relation exist. For instance, an aggregated property is whether a person knows anybody who attends a certain school.

Here we use the normalized discounted cumulative gain (NDCG) [6] to evaluate a predicted ranking, which is calculated by summing over all the gains along the rank list R with a log discount factor as NDCG$(R) = Z \sum_k (2^{r(k)} - 1/\log(1+k)$, where $r(k)$ denotes the target label for the k-th ranked item in R, and Z is chosen such that a perfect ranking obtains value 1. The better an algorithm, the higher would the friendship test tuple be ranked.

5.2 Results

In the experiment, we analyzed to what degree friendship variables (i.e., the relation *Knows*) can be predicted from the friendship patterns (the patterns in the friendship columns) alone and to what degree person's attributes and aggregated attributes improve the predictive performance. Also we were interested to see to what degree the setting of the thresholds ϵ in the pruning steps affected the results. All experiments were done with 2,000 persons in the training set and 2000 persons in the test set. All experimental results were repeated with 5 non-overlapping sets to obtain confidence values.

The Table 1 shows results when the threshold for the *Knows* relation was set to 10, i.e., columns of friends with less than 10 entries were removed. The columns in the table are labeled by the threshold chosen for the attributes (first number) and the aggregated attributes (second number). We chose different numbers for attributes and aggregated attributes since there were many fewer aggregated features than attributes. Table 2 shows results where the threshold for the *Knows* relations was set to 20. We only report results for LDA and for SVD; the results for NNMF were consistently worse than the results for those two algorithms. The number of topics / principle components was chosen to be optimally: about 100 in all cases.

The first thing to observe is that the second experiment resulted in better scores. This can be explained by the fact that there is stronger correlation between friendship patterns of popular friends. As can be seen in the tables, attributes and aggregated attributes improved the score significantly for SVD. For SVD, a *zero* threshold for both attributes and aggregates features gave overall best results (i.e., no pruning). Interestingly, when all attributes and aggregated attributes were included, the results for the inductive setting were slightly better than the results for the transductive setting. LDA could only beat SVD when no attributes were used (threshold infinity). Interestingly, the performance of LDA was not improved in general when attributes or aggregated attributes were used.

6 Discussion and Conclusions

We have presented a learning approach for relational data. The learning process is based is to a large degree autonomous. Mainly the statistical unit and the population need to be defined by a user. Since the number of columns in the data matrix is, to a first approximation, independent of the overall size of the RDB and since the sample size can be controlled, training time is essentially independent of the overall size of the RDB. The generalization from the sample to the population is linear in the size of the population.

In our experiments based on the FOAF data set, SVD showed most consistent performance. Both LDA and SVD exploited the benefits of multivariate prediction since approaches based on single predictions (not reported here) did not even reach the performance of very simple benchmark approaches.

The tuples with their certainty values can be entered into the RDB. Although the number of estimated tuples is much smaller than the number of possible tuples, in many cases one might want to apply an additional selection process, e.g., only store the N tuples in a relation with highest probability. The derived probabilistic tuples can readily be integrated into SQL queries. For example, based on the learnt results using the FOAF data set, one could answer queries such as: *Who would likely want to be Jack's friend; which female persons in the north-east US, would likely want to be Jack's friends.*

The approach relies on the fact that, in the relational graph, information is local to a statistical unit. Let's assume that any descendant of Rockefeller is rich. If the data base only contains the relation *ChildOf*, the presented approach could not directly learn that rule. On the other hand, the father or mother of a descendant of Rockefeller would also be rich (at least one of them is a descendant of Rockefeller as well). Also a descendant of Rockefeller might typically have rich friends. The presented approach would be able to exploit this information and conclude that a rich parent and rich friends would be indicator of wealth.

For an analysis of statistical inference one can distinguish three different scenarios. First, we consider a transductive setting, where we only consider statistical units in the sample and predict the truth values of their tuples. This corresponds to inference used in many recommendation systems (e.g., the Netflix competition). Second, we generalize the learnt model to statistical units in the population (i.e., the RDB) that are outside the sample and not observed during the training. Considering the design of the experiment where we generated the sample as a random subset of statistical units in the data base, this type of inference is sound. Third, we consider the generalization to statistical units which are not in the RDB. This type of generalization is of course the most interesting one but also the most difficult one to analyze. This latter type of generalization is an area of active research in network analysis.

Acknowledgements: We acknowledge funding by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project and by the EU FP 7 Large-Scale Integrating Project LarKC.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3, 2003.

[2] D. Brickley and L. Miller. *The Friend of a Friend (FOAF) project*. http://www.foaf-project.org/.

[3] L. De Raedt. Attribute-value learning versus inductive logic programming: The missing links (extended abstract). In *ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming*. Springer-Verlag, 1998.

[4] P. Domingos and M. Richardson. Markov logic: A unifying framework for statistical relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[5] L. Getoor, N. Friedman, D. Koller, A. Pferrer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[6] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR'00*, 2000.

[7] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *Poceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[8] W. Kent. A simple guide to five normal forms in relational database theory. *Communications of the ACM*, 26, 1983.

[9] LarKC. *The large Knowledge Collider*. EU FP 7 Large-Scale Integrating Project, http://www.larkc.eu/, 2008.

[10] N. Lavrač, S. Džeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *EWSL-91: Proceedings of the European working session on learning on Machine learning*, 1991.

[11] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999.

[12] C. Lippert, S. H. Weber, Y. Huang, V. Tresp, M. Schubert, and H.-P. Kriegel. Relation-prediction in multi-relational domains using matrix-factorization. In *NIPS 2008 Workshop: Structured Input - Structured Output*, 2008.

[13] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*. Ohmsma, Tokyo, 1990.

[14] A. Popescul and L. H. Ungar. Feature generation and selection in multi-relational statistical learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[15] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3), 1990.

[16] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[17] V. Tresp, Y. Huang, M. Bundschus, and A. Rettinger. Materializing and querying learned knowledge. In *Proceedings of the First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web*, 2009.

[18] V. Tresp and K. Yu. Learning with dependencies between several response variables. In *Tutorial at ICML 2009*, 2009.

[19] Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. Infinite hidden relational models. In *Uncertainty in Artificial Intelligence (UAI)*, 2006.