Predictive Modeling using Features derived from Paths in Relational Graphs

Yi Huang (1), Volker Tresp (2), Stefan Hagen Weber (2)

(1) Ludwig-Maximilian University Munich, Germany
(2) Siemens AG, Corporate Technology
Information and Communications
Learning Systems, Munich, Germany

Technical Report as a Deliverable in the THESEUS Project

June, 2007

Abstract

This paper is concerned with supervised learning in relational domains. As relational framework, we use the Resource Description Framework (RDF) that is the basis for representing information about resources in the World Wide Web. The fundamental RDF structure is a relational graph such that feature derivation can be formulated in a simple graphical context. We present learning solutions for literal prediction, classification and relation prediction. We discuss feature-based learning and kernel-based learning. We present experimental results using a bio-informatics data set and a recommendation data set.

1 Introduction

Due to the increasing relevance of interconnected networked domains, relational learning is an area of growing interest. Past approaches were often based on relational data bases [5], entity relationship models [8] or first-order logic [4, 14, 3]. A more recent relational representation is the Resource Description Framework (RDF), which is a simple scalable general-purpose syntax for representing information about resources that can be identified on the World Wide Web [11], and is considered the basis for the Semantic Web [11]. RDF is also a useful model for other relational domains and might become the basis for a general information framework, similarly as XML is for simply structured data. Thus it can be expected that an increasing amount of data will be offered in the form of RDF statements; large RDF databases for RDF triples have already been developed.

In RDF, the central structure is a labeled connection between two resources, i.e. the RDF-triple of the form (s, p, o) where s, p, o stands for subject, predicate (or property), and object. Based on a set of triples, a complete knowledge base can be represented as a directed (potentially cyclic) graph where subjects and objects are nodes and a property is a directed labeled edge from subject to object. Thus relational queries and relational learning can be formulated in the context of a very simple scalable structure, i.e., the RDF-graph, which is an important advantage if compared to other more complex representations. Higher order relations are handled by introducing *blank nodes* thus reducing higher order relations to simple triples. Every resource is an instance of one class or potentially several classes as specified by *type* properties. Furthermore, classes are organized into class hierarchies. Since properties are defined as classes, they might be organized hierarchically, e.g. closed friend might be a subproperty of friend. The RDF vocabulary description language is RDF-Schema (RDFS) and describes a detailed set of classes, subclass relations, properties and sub-property relations of a given domain. Furthermore, RDFS defines the type constraints of the subject (domain) and the object (range) of RDF-triples. Note that certain triples are entailed by RDFS such that if a resource belongs to a class via the *type* property then it is also belongs to each parent class, again via a *type* property. Entailment can be implemented by adding these triples explicitly. In this paper we assume that all entailed triples have been added. A very interesting feature is that both RDF and RDFS form one joint graph: RDFS is also RDF. Figure 1 shows an excerpt of a simple RDF graph example.

Every RDF node is either a resource with a unique identifier or a constant, called a literal.[1] In this paper a resource will always stand for a resource with a unique identifier. Character string literals are plain literals whereas typed literals might be floats, booleans, integers, etc.. A subject must always be a resource whereas an object in an RDF triple can either be a resource or a constant.

RDF is the basis for the Semantic Web and learning in RDF-descriptions

[1]We will not describe the RDF-formalism explicitly but will simply assume that resource with a unique identifier can uniquely be identified.

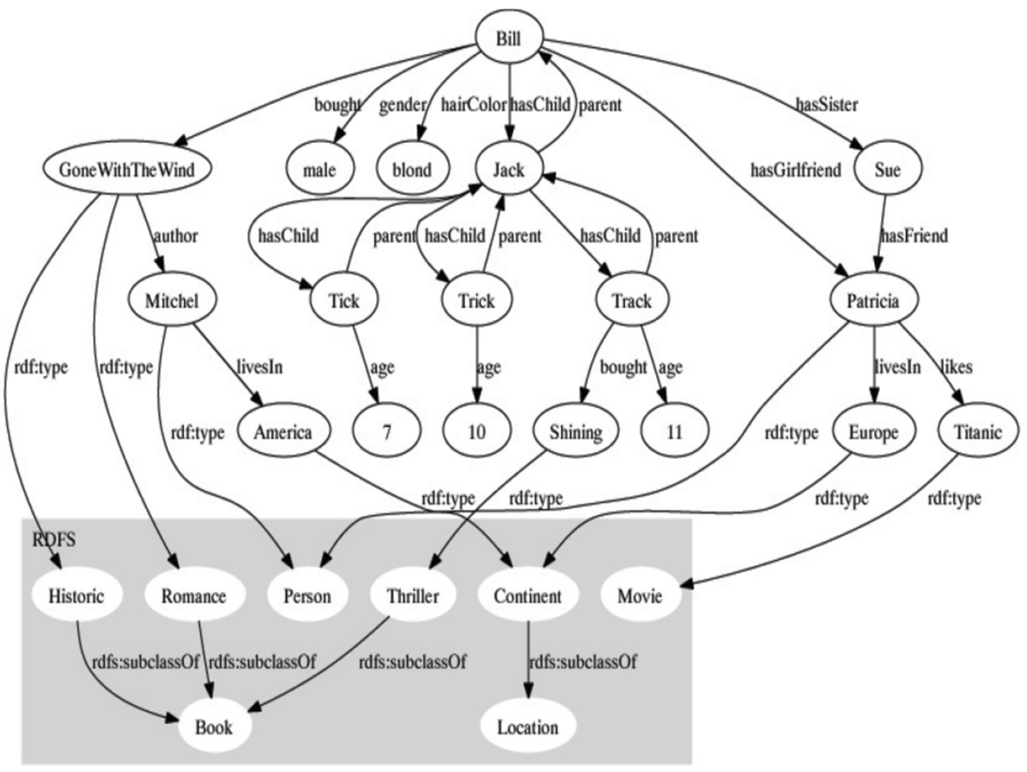

Figure 1: Example of an RDF-Graph excerpt describing relatives and interests of Bill. Note that the lower part shows RDFS.

might become increasingly relevant. The fact that an RDF knowledge base is represented in form of a directed graph makes this representation especially suitable for learning, in particular for extracting meaningful features (see Section 2). There are many potential learning tasks. In this paper we consider two learning tasks. The first one considers the prediction of the state of typed literals with classification as a special case. The second case consists of the prediction of the existence of an RDF-triple where subject and object are resources. In contrast to other approaches [5, 8, 3] the focus here is supervised learning. Furthermore, the focus is predictive accuracy but not explanation as in many other approaches to relational learning [13].

The paper is organized as follows. In the next section we will consider literal prediction and in Section 3 we will discuss relational prediction. In Section 4 we discuss suitable learning approaches. Section 5 discusses related work and Section 6 describes our experiments. In Section 7 we present conclusions.

2 Literal Prediction

2.1 Boolean Literals

First, we consider the case that all literals are boolean.[2] If in the triple (s, p, o) o is a boolean literal, we can simplify this triple to (s, p) and remove the boolean literals from the RDF-graph.[3] The corresponding property is denoted as a boolean property. The non-existence of a statement will imply that the boolean property is not true (closed-world assumption).

Now we consider the case that we want to predict the probability that a particular resource s_i possesses the target boolean property p_j. We assume that the RDF-schema (RDFS) as well as the type of each resource are known. Also note that in the following we operate in the RDF-graph and that we assume that all triples entailed by RDFS have been added. In general, we will deal with one connected RDF-graph (the *domain graph*) but it is also conceivable that the RDF-graph can be partitioned into several disconnected graphs. In the following we will assume one connected graph. A test case concerns the prediction of the existence of one unknown boolean literal in the RDF-graph.

We make the following modeling assumption: the existence of a boolean literal p_j can be predicted solely based on features derived for the subject s_i of that literal

$$p_j | \mathcal{F}_i$$

where $\mathcal{F}_i$ is a suitable set of features describing the subject s_i. In a given application, this dependency $p_j | \mathcal{F}_i$ might be modeled by any suitable probabilistic or non-probabilistic classifier (see Section 4).

We now consider the generation of the training set. Since we assume that the RDFS is known, the domain for the target property p_j is known. The training set is defined by all resources in the RDF graph that belong to the specified domain excluding the test case.[4] These resources define the training instances.

The most challenging problem is now to generate suitable features. In a preprocessing step, we add to each triple the inverse triple, if it does not

[2]Note, that there are a number of ways of how a non-boolean literal (discrete, continuous) can be turned into a set of boolean variables. As an example, *(Person, hairColor, color)* might be turned into a boolean representation by replacing it with *(Person, hasBlondHair)*, *(Person, hasBlackHair)*, etc.. For plain literals, e.g., the name of a person, we only indicate if that particular plain literal exists or not but don't represent its content.

[3]Or introduce a dummy object node to maintain RDF-conformity.

[4]Application specific, we might reduce that set.

yet exist. Thus, if (s, p, o) is a triple and if o is not a literal, we add the triple $(o, \bar{p} = inverse(p), s)$. Note that $(s, inverse(\bar{p}), o) = (s, p, o)$. Due to its simple structure, introducing an inverse relation is straightforward in the RDF-graph but much more involved in other relational representations.

Let's consider a particular training instance s_i. As it is a typical assumption in other domains with connected data, e.g., time-series problems or image processing, we assume that primarily a local environment of s_i in the graph will have predictive power. Consequently, we will derive features which describe the local environment of instances in the training set. In a first step, starting from s_i (start node) we generate all possible paths up to a predefined length K, where K is an important tuning hyperparameter.[5]. In graph theory, a path in a graph is a sequence of nodes such that from each of its nodes there is an edge to the next vertex in the sequence. The length of a path K is the number of edges that the path uses. We define a *legal* path *lpath* to be a tuple constructed by all properties and resources collected on the path, thus the k-th path for subject s_i

$$\text{Lpath}_{i,k} = (s_i, p_{i,k,1}, r_{i,k,1}, p_{i,k,2}, r_{i,k,2}, ..., r_{i,k,l-1}(, p_{i,k,l})) \quad l \leq K$$

where $r_{.,.,.}$ is a resource on the legal path and where $p_{.,.,.}$ is a property transversed from subject to object, i.e., following the direction of the arc, and might be an original property or an inverse property as inserted previously. Note, that a legal path can be terminated by a resource or by a property. We apply the following two restrictions on a legal path. First, a legal path cannot reverse: thus if (s, p, o) is in the legal path, then $(o, inverse(p), s)$ cannot. Second, a particular (s, p, o) cannot be included more than once. Finally, we apply the following procedure, if a cycle is encountered: If a resource is encountered that is already present on the legal path, then another copy of the legal path is added to the set and is terminated by an inserted property of the form $equal(m)$ where m is the position in the legal path where the identical resource was encountered the first time. Note that all prefix legal paths are also legal paths.

For each training instance s_i we now introduce new *derived boolean properties* $\tilde{p}$, which come in two types. Type 1 corresponds to $\tilde{p}_{i,k} = p_{i,k,1}, p_{i,k,2}, ..., p_{i,k,l}$ and represents the concatenation of all properties in a legal path. Type 2 corresponds to the case that the last element in a legal path is a resource r_m and is written as $\tilde{p}_{i,k,m} = p_{i,k,1}, p_{i,k,2}, ..., p_{i,k,l}, r_m$. Now a boolean RDF statement $(s_i, \tilde{p})$ is introduced for each $\tilde{p}$ that could be derived from a legal path starting at s_i. Thus we have reduced graphical information local to s_i to a set of simple boolean properties of s_i!

[5]Application specific, we might chose a different criteria.

It is now simple to define the feature set. Consider the set of all $\tilde{p}$ generated from all training instances and potentially the test instance. For each distinct $\tilde{p}$ we introduce a feature; *the value of that feature in training instance s_i indicates how often that $\tilde{p}$ exists for that s_i.*

Note, that we don't represent all possible paths as feature variables but only those actually present in the RDF-graph.

The procedure allows us to generate a number of interesting and expressive features. Here are some examples for derived boolean properties for a subject s_i: *(hairColorBlond); (bought,GoneWithTheWind)*[6]*; (hasChild, Jack, hasChild) (3 times); (hasChild,Jack); (sister,friend,boyfriend, equal(1))* This describes a blond-haired person who bought the book titled "Gone With the Wind", who has one child named Jack, Jack has 3 children, the initial person has a sister who has a friend and the boyfriend of this friend is the person (subject).

2.2 Discussion: Complex Features, Feature Selection, Postprocessing, and Complexity

Note that we have derived local properties $\tilde{p}$ but we have not derived more complex logical sentences (for example, by AND-ing properties) as it is done by FOIL-based learning approaches [9, 10]. The search for suitable logical sentences can be time-consuming; our philosophy is that this step, if necessary, should be the task of the learning machine using the features.

The most obvious tuning parameter is K, the maximum length of the paths to be explored. Features that are nonzero only once (resp. a minimum number of times) can naturally be removed. Similarly it might make sense to remove complete feature classes. Naturally, features can be further processes, for example binarized by thresholding, etc..

With a branching factor upper bounded by N the number of possible Type 1 derived boolean properties $\tilde{p}$ is upper bounded by N^K. Thus in problems suitable for our approach, both numbers should not be too large. With L objects in the range-type we obtain $N^K \times L$ possible features derived from Type 2 properties. Since L is often large, care has to be taken not to include features derived from uninformative Type 2 derived boolean properties $\tilde{p}$.

2.3 Non-boolean Literals

If the target property is a non-boolean typed literal then we simply have to select a machine learning approach being able to deal with such targets, e.g.,

[6]URI to the book resource with the title "Gone With the Wind"

continuous, multi-valued, ... targets.

For the non-target non-boolean literals, i.e., when non-boolean typed literals appear on legal paths, an appropriate form of aggregation might be used.

Consider the derived properties *(Jack, child, age, 7), (child, age, 10), (child, age, 11), (child, age, 15)* The aggregated features might now be *(child, averageAge, 14.3) and (child, minimumAge, 7).*

For plain literals (strings) we only indicate if that particular plain literal exists or not.

3 Relation Prediction

We consider now an RDF-triple (s_i, p_j, o_k) where o_k now is a resource (and not a literal, as in the last section). The prediction problem is that given two resources s_i and o_k, what is the probability that the triple (s_i, p_j, o_k) is present.

3.1 One-sided Prediction

Consider a target triple (s_i, p_j, o_k). We might define a specific boolean property $p_{j,k}$ for each training object under consideration, similar as we have done in the case of Type 2 derived property $\tilde{p}$ in the last section. Only here we perform that operation on the target triple. Having this target boolean property the methods described in the last section can be applied. As an example consider triples of the form $(person, bought, book)$. Dependent on the book to predict, the new target boolean property might be then $(person, boughtGoneWithTheWind)$, $(person, boughtShining)$, etc.. Note, that we have turned one classification problem into many simpler ones.

3.2 Two-sided Prediction

One-sided prediction might not always be suitable. Again, we consider an RDF-triple (s_i, p_j, o_k) where o_k now is a resource. The prediction problem is that given two resources s_i and o_k, what is the probability that the triple (s_i, p_j, o_k) is present.

We assume a model where this probability is independent of all other evidence given features describing the involved resources s_i and o_k and features derived from their mutual relation and thus is modeled as

$$(s_i, p_j, o_k)|\mathcal{F}_i, \mathcal{F}_k, \mathcal{F}_{i,k}$$

Now consider the generation of the training set. Considering the property p_j, the training set consists of all pairs s_i, o_k where s_i is in the domain of p_j and o_k is in the range of p_j. Note, that the train set size is typically much larger than the training set considered in one-sided prediction. $\mathcal{F}_i$ and $\mathcal{F}_k$ are features generated for s_i resp. o_k as described in the last section.

Now consider a legal path originating from s_i and terminating at o_k (or *vice versa*). In this case this legal path is removed from the set of paths originating from s_i and is added to the set of *joint relational legal paths.* The terminal resource o_k is replaced by the property *equalToObject*. Equivalently to before a corresponding boolean property $\tilde{p}$ is introduced and a feature is added to $\mathcal{F}_{i,k}$ which is true when $(s_i, \tilde{p})$ exists.

The representation of joint features $\mathcal{F}_{i,k}$ permits the learning of the property $(s_i, grandFather, o_k)$ using the path (starting from s_i) *(parent, parent, equalToObject)*. The learning machine has to AND the corresponding feature of o_k with boolean literal: $(male)$.

4 Learning Algorithms

4.1 Literal Prediction and One-sided Relational Prediction

For literal prediction, any appropriate machine learning approach that can handle the typically large feature set can be used. E.g., if the literal is boolean, a binary classifier might be appropriate. Needless to say that the same applies to one-sided relational prediction.

4.2 Two-sided Relation Prediction

Relation prediction can be performed by any suitable binary classifier. Recall that the feature set and the training data set will be considerably larger than for one-sided prediction. For example, if R is the number of features for s_i resp. o_k, then in the two-sided approach we have $2R$ features and up to R^2 training instances. Thus typically we can only apply learning machines, which can handle large feature sets and large training sets.

Simple classifiers might often have problems to exploit clustering structure in the data such as that some females might prefer romantic movies and some males might like action movies. Only the nonlinear combination of the features of movies and users will produce an appropriate feature representation. To facilitate learning we might want to pre-structure the features. If a kernel-based approach is chosen, we might select as appropriate kernel

$$k((s_i, o_k), (s_{i'}, o_{k'})) = k_s(s_i, s_{i'})k_o(o_k, o_{k'}) + k_{s,p}(s_i, s_{i'}, o_k, o_{k'}) \tag{1}$$

where $k(s_i, s_{i'})$ is a kernel based solely on $\mathcal{F}_i$, $k_o(o_k, o_{l'})$ a kernel solely based on $\mathcal{F}_k$, and where $k_{s,p}(s_i, s_{i'}, o_k, o_{l'})$ is a kernel solely based on $\mathcal{F}_{i,k}$. A kernel product has a conjunctive character. [7]

5 Related Work

Due to the novelty of the RDF framework there has been little work on relational learning using RDF-graphs. There is some recent activity on learning on the Semantic Web [16], but the focus so far has been mostly on document annotation, ontology learning and content management. Considering that we are essentially solving learning tasks in relational domain, our work is closely related to inductive logic programming (ILP) [4]. Although classically concerned with the learning of rule sets (a classical paper here is [13]), recent extensions have combined the rule selection algorithms of ILP algorithms with statistical classification [9, 10, 2, 6]. Those approaches have advantages if the dependency to be learned is close to deterministic, requires an explanation component and is potentially complex and justifies extensive search, whereas we see advantages in our approach in more statistical settings where the prediction can only be done with considerable uncertainty and explanation of dependencies is not of prime importance. The work on Statistical Relational Learning (SRL) [7, 8, 3, 12] is also quite related but focusses on the development of joint probabilistic models whereas we focus on supervised learning. One of the simplest interesting relational domains is collaborative filtering with resources users and items. Most approaches perform what we have coined one-sided relational prediction. One of the few two-sided approaches we are aware of is described in [17]. That approach is limited to considering local properties and relations and requires expensive learning of the kernel in a hierarchical Bayesian framework.

6 Experiments

We evaluated our approach using two data sets. The first one concerns a movie recommendation system. Recommendations are made based on pre-

[7]Alternatively in a feature-based approach we might want to add product features to the representation, i.e., if $f_{i,m}$ and $f_{k,n}$ are features for subject and object, we add the feature $f_{i,m}f_{k,n}$. Recall that a product of kernels corresponds to the mutual product of all features.

vious ratings and user and movie attributes. The second one concerns the prediction of gene functions based on gene attributes and gene interactions. We performed experiments with various classifiers. The classifiers are: naïve Bayes classifier (NB), logistic regression (LogR), support vector machines (SVMs, implemented using the SVM^{light} package), Gaussian process classification (GPC) and kernel smoothing (ksm). For the latter three methods various kernels were tried: a linear kernel (lin), a cosine kernel (cos), a Gaussian RBF-kernel (RBF), a second degree polynomial kernel (poly), and an exponential linear kernel (expl). The latter is defined as $k(x_i, x_j) = \exp Ax_i^T x_j$, where $A > 0$ is a tuning parameter. In the experiment the results were not very sensitive to A and we selected $A = 0.2$.

6.1 Recommendation System

The first data set we used is MovieLens [15]. We binarized the ratings and generated RDF triples of the form $(user, likes, movie)$. In addition there are triples with typed literals describing attributes of users and movies such as age, $gender$ and $occupation$ for users and $publishedYear$, $genres$ and so on for movies. All attributes were represented as boolean variables. We selected a subset with 156 users, 603 movies and 17,336 ratings, such that each movie has at least 30 ratings and each user rated more than 6 movies. We report average accuracy on 10 different test sets where each test set contains one unknown movie rating from each user. For each user the following properties were derived: $(young)$, $(mediumAge)$, ..., $(female)$, ... for user attributes, and $(likesMovie_i)$ for all movies. Finally, $(likes, MovieAttr)$ aggregates the attributes of the movies that a user likes. For example, the feature derived from $(likes, action)$ aggregates how many action movies a user likes. Equivalently, for each movie features are generated from movie attributes, from the identity of the users who liked the movie and from their aggregated attributes (e.g., how many older persons liked the movie, ...).

We performed experiments with one-sided and two-sided relational prediction. For user-sided prediction a separate model is trained for each movie (603) and for movie-sided prediction, a separate model is trained for each user (156). In two-sided relational prediction, only one global model is trained. For kernels, the product kernel in Equation 1 was used.

Table 1 shows averaged test set accuracy. If one compares the two one-sided approaches, then movie-sided prediction is superior to user-sided prediction. In the user-sided prediction, previous ratings are more informative than user attributes. Performance cannot be improved by adding user attributes or aggregated movie attributes. The exception is the SVM where we see improvements for two kernels. In movie-sided prediction, also past

Table 1: Experimental results (accuracy) on the recommendation data set.

	user-sided				movie-sided				two-sided		
	U	R_u	R_u+U	R_u+U+M	M	R_m	R_m+M	R_m+M+U	M+U	$R_{u,m}$	$R_{u,m}$+M+U
NB	55.5±0.4	59.5±1.1	59.2±1.1	59.2±1.1	58.5±1.1	58.8±0.8	59.1±1.0	62.1±0.8	56.5±0.6	61.7±1.0	61.3±0.9
SVM, lin	59.1±0.8	63.4±1.1	62.3±1.0	61.7±1.0	59.4±1.0	61.9±1.0	61.5±0.8	62.1±1.1	N/A	N/A	N/A
SVM, RBF	59.5±0.9	61.3±1.0	61.3±1.0	61.2±1.1	61.3±1.0	63.5±1.2	63.9±1.2	64.1±1.1	N/A	N/A	N/A
SVM, poly	56.8±1.0	59.5±1.3	60.8±1.2	62.2±1.3	60.2±0.8	62.1±0.8	62.4±1.0	65.3±1.0	N/A	N/A	N/A
SVM, expl	55.9±0.9	59.7±1.3	60.8±1.2	60.0±1.3	52.6±1.1	54.8±0.7	56.3±1.2	61.4±1.1	N/A	N/A	N/A
ksm, lin	60.8±1.2	64.6±1.1	62.3±1.5	62.7±1.2	62.9±1.0	64.6±1.1	63.6±1.4	66.3±1.0	60.8±1.2	62.8±1.3	57.8±1.2
ksm, cos	60.9±1.3	65.7±1.3	63.3±1.4	62.2±1.4	63.0±1.3	64.6±1.2	64.4±1.4	66.6±0.8	57.8±1.2	62.8±1.1	57.8±1.2
ksm, expl	61.0±1.2	63.3±1.2	63.6±1.3	62.2±1.5	63.1±1.2	65.2±1.2	65.5±1.2	66.9±1.1	59.5±1.0	66.4±1.0	64.7±1.2
GPC, cos	61.1±1.2	64.8±1.1	62.4±1.5	62.8±1.2	63.1±1.4	64.9±1.1	63.6±1.4	66.2±1.0	N/A	N/A	N/A
LogR	61.1±1.2	65.2±1.2	63.0±1.4	62.7.7±1.3	62.6±1.1	65.2±1.2	65,1±1.4	67.6±1.0	59.6±0.9	65.5±1.3	65.6±1.0

rating information is more informative than movie attributes. Adding movie attributes to the rating information in most cases does not lead to an improvement. Quite surprising is that adding aggregated user information improves performance significantly. For the two-sided experiments, again the rating information is more relevant than movie and user attributes. If we compare systems with only relational information, the kernel smoother with the exponential linear kernel performs best. Adding attribute information does not improve performance in two-sided prediction. When we compare methods, logistic regression performs best, followed by the kernel smoothers, SVM and naïve Bayes. N/A experiments were not performed due to run-time problems.

It is instructive to compare our approach and our results to previous approaches to recommender systems. Most previous approaches are systems which are purely rating based, either user-sided or movies sided. Some researchers have added content information, user attributes when user-sided and movie attributes when movie-sided. As in our experiments adding this information does not, in general, improve performance. Surprisingly in our experiments is that adding aggregated user attributes to movie-sided predicting boosts performance. We believe that this is quite an interesting outcome of our general approach to relational modeling. Our approach to two-sided prediction is novel. Although so far not the overall best approach, it is interesting to note that the product kernel is quite effective for kernel smoothing in two-sided prediction.

6.2 Gene Data

We briefly want to report ongoing work on the analysis of gene data set of KDD Cup 2001 [1]. A gene is described by a number of attributes including *chromesome*, *essential*, *phenotype*, *motif*, *class* and *function*. In addition genes can interact leading to triples where both subject and object are genes. Our task is to predict functions of genomes. Totally 1243 genes are contained in the data set and split into training set (862) and test set(381). The challenge of the data is that only a small number of gene interaction relations are known. In one set of experiments, gene functions where treated as independent attributes to be predicted. Best performance here could be achieved using a SVM with RBF kernel based on gene attributes, gene interactions and aggregated interactions achieving an accuracy of 80.5 %. Much better results were achieved by treating the functions as resources and using two-sided prediction. With product kernels based on exponential linear kernels, we achieved an accuracy of 93.2 %. This large improvement can be explained by the collaborative effect between genes and functions, which means that unknown functions can be predicted from other known functions,i.e., functions are partially correlated.

7 Conclusions

We have presented a novel approach to learning in relational graphs. We focussed on literal prediction, classification and relation prediction. The critical problem in supervised relational learning is the generation of appropriate features. Our proposed method for feature generation is straightforward to apply and does not involve complex search procedures. The experimental results demonstrate that the presented approach gives competitive performance in challenging problems. Many well known approaches, e.g., many learning approaches for recommender systems, are special cases of our approach putting them into a larger framework. Our approach can motivate novel features as the aggregated user attributes in our experiment. Our approach puts two-sided relational prediction into a novel framework, which needs to be explored further. We believe that our experimental results on the movie recommendation system and on the genomic data are quite interesting.
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] Jie Cheng, Christor Hatzis, Hisashi Hayashi, Mark-A. Krogel, Shinichi Morishita, David Page, and Jun Sese. KDD cup 2001 report. *SIGKDD Explorations*, 3(2):47–64, 2002.

[2] J. Davis, I. Ong, J. Struyf, E. Elizabeth Burnside, D. David Page, and V. Santos-Costa. Change of representation for statistical relational learning. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[3] P. Domingos and M. Richardson. Markov logic networks. *Machine Learning*, 62, 2006.

[4] S. Dzeroski and N. Lavrac, editors. *Relational Data Mining*. Springer, Berlin, 2001.

[5] N. Friedman, D. Getoor, L. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. 16th IJCAI*, pages 1300–1309. Morgan Kaufmann, 1999.

[6] T. Gaertner, J.W. Lloyd, and P.A. Flach. Kernels and distances for structured data. *Machine Learning*, 2004.

[7] L. Getoor, D. Koller, and N. Friedman. From instances to classes in probabilistic relational models. In *Proc. ICML 2000 Workshop on Attribute-Value and Relational Learning*, 2000.

[8] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft, 2004.

[9] N. Landwehr, K. Kersting, and L. De Raedt. nfoil: Integrating naive bayes and foil. *Journal of Machine Learning Research*, 2007.

[10] N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. kfoil: Learning simple relational kernels. *AAAI 2006*, 2006.

[11] F. Manola and E. Miller. *RDF Primer*. W3C Recommendation.

[12] J. Neville and D. D. Jensen. Dependency networks for relational data. *Proceedings of the Fourth IEEE International Conference on Data Mining*, 2004.

[13] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5, 1990.

[14] Luc De Raedt and K. Kersting. Probabilistic logic learning. *SIGKDD Explor. Newsl.*, 5(1):31–48, 2003.

[15] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[16] S. Staab. Position statement: An overview on machine learning for the semantic web. *Dagstuhl Seminar on Machine Learning for the Semantic Web*, 2005.

[17] K. Yu, W. Chu, S. Yu, V. Tresp, and Z. Xu. Stochastic relational models for discriminative link prediction. *NIPS 2006*, 2006.