Aggregating Predictions vs. Aggregating Features for Relational Classification: A Comparison

Oliver Schulte and Kurt Routley

School of Computing Science, Simon Fraser University,
Burnaby, B.C., Canada V5A 1S6, oschulte@cs.sfu.ca,kdr4@sfu.ca

Abstract. This paper presents a fast, principled, and accurate method for leveraging standard machine learning classifiers for relational classification. The method uses the average of a classifier score over a target's relational neighbourhood as an aggregate classifier score. We show that the average is consistent with the classic random selection semantics for probabilistic logic. Our experiments compared different relational classifiers on sports data. Compared to propositionalization methods and other score aggregators, the random selection classifier showed robust and competitive performance.

1 Introduction

Most real-world structured data are stored in the relational format, with different types of entities and information about their attributes and links between the entities. Relational data classification is the problem of predicting a *class label* of a target entity given information about features (attributes) of the entity, of the related entities, or neighbors, and of the links. This paper presents a fast, principled, and accurate method for leveraging standard machine learning classifiers for relational classification.

A key challenge for relational classification is that the number of links of the target entity is not uniformly bounded. Since the features of each neighbor potentially carry information about the target class label, the number of predictive features for classification is thus a function of the size of the target entities neighborhood, rather than a fixed dimensionality d. Relational classifiers therefore aggregate the information from the target entity's neighborhood. There are two fundamental options for aggregation: 1) First aggregate the neighbors' features into a single aggregate feature vector, then classify based on the aggregate vector. 2) First derive a classification score based on a single neighbor, then aggregate the scores. In this paper we compare the two approaches empirically on data sets with continuous features. Since standard relational benchmark datasets contain discrete features mainly or only, we use two real-world continuous datasets that summarize players' actions in ice hockey and in soccer. The ice hockey dataset was obtained by a web crawler and has not been analyzed before.

Computionally, classifier training with score aggregation can be done very simply by forming a data table such that one row contains the features of one neighbor, and applying a regular non-relational learning algorithm to this table.[1]

Evaluation. We use standard aggregation functions to aggregate continuous features (mean, sum, min, max). Once features have been aggregated, any standard single-table machine learning classifier for continuous features can be applied for classification. In this paper we apply logistic regression and support vector machines (SVMs).

Both logistic regression and SVMs return a continuous classification score. We apply the single-table classifier to the features of each neighbor to obtain a classification score for the neighbor, then aggregate the scores. For aggregating scores, we use the same functions as for aggregating features. In addition we apply noisy-or, a standard rule for combining a list of probabilities into a single probability.

The basic task is to predict the result of a given target team in a given target match (win or not). We examine hockey data from the NHL and soccer data from the UK Premier League. The training set contains data from previous matches and the test set data from the following ones. We experiment with two different feature sets: First, summary statistics from the previous season for each player (e.g., number of goals scored by the player). Second, in addition, the action counts for each player on the target team in the target match.

Our main conclusion is that among score aggregation operators, the average or mean provides robust competitive performance across different settings. We show that the average operator is consistent with the classic random selection semantics of probabilistic 1st-order logic due to Halpern and Bacchus [10, 4]. The performances of the best score and feature aggregation methods is close except for one dataset where feature aggregation is clearly superior. We outline a hybrid approach that combines informative aggregate features with score-based classification.

Contributions. Our main contributions may be summarized as follows.

1. The first direct comparison of feature aggregation vs. score aggregation methods for relational classification.
2. A new score aggregation method that uses the expected classification score from a random selection of the target node's neighbour.
3. A new real-world dataset in relational database format with play summaries from 3,857 NHL matches, available on-line from `ftp://ftp.fas.sfu.ca/pub/cs/oschulte/hockey`.

2 Related Work

Because of the importance of relational data, there has been much work on link-based classification. For overviews please see [6, 5]. We provide a high-level

[1] If the neighbourhood sizes of different target entities differ, a simple adjustment of the classifier loss for neighborhood size is necessary, see details below.

description of the work most relevant to the question of feature vs. score aggregation.

Aggregating Classifier Scores. Most approaches that aggregate classification scores use a function that maps a list of probabilities to a single probability. Following the terminology of Bayes nets, such functions are referred to as *combining rules* [18, 12]. In our terminology, a combining rule is a special kind of classifier score aggregation. While the arithmetic mean of class probabilities has been considered for aggregating probabilities in relational learning [17], the geometric mean is a new proposal motivated by the random selection semantics. The random selection semantics is a classic concept from AI research that combines 1st-order logic with probabilities [10, 4]. The key idea is to interpret a free 1st-order variable as a random variable that randomly selects an element from its domain. The random selection semantics was previously applied for learning generative Bayes net models [21]. To our knowledge we present the first application to relational discriminative learning.[2]

Propositionalization. The majority of work on relational classification has adopted the feature aggregation strategy. This approach of "flattening" the relational structure is generally known as *propositionalization* [13]. For continuous features, propositionalization methods use the same standard aggregate functions that we use in this paper [14, 25]. For discrete data, a common approach is to use a *feature function.* A feature function maps a relational neighbourhood to a single value for the given feature. For instance, if the feature is "student's grade is A", the count feature function returns the number of A's achieved by the student. If the feature function returns a continuous or integer value, the values of the feature functions are used as inputs to a log-linear model (conditional random field) for prediction [23, 24, 8, 16]. A feature function may also return a discrete value; a commonly used binary feature function is existential quantification, for example using 1 as a classifier if the student has achieved an A in some course, and a 0 otherwise.

Complex Features. The most expressive propositionalization models apply feature functions to combinations of the discrete features given in the data (e.g., [15]). For instance, to predict the ranking of a student, we may distinguish the number of A grades achieved in higher-level course from those achieved in lower-level courses. Complex discrete features may be combined with aggregation functions, as aggregation conditions, for continuous variables [25, 20]. For example, to predict the age of a user in a social network, we may consider the average age of her friends who have the same gender and live in the same city.

Several researchers discuss advantages and disadvantages of propositionalization for link-based classification [7, 6]. The main advantage is expressiveness: feature generation methods search a large space of potentially useful features. If an

[2] A related previous paper was presented at the 2012 UAI-StarAI workshop. The nonarchival workshop did not publish proceedings.

informative new complex feature or aggregate feature can be found, it improves classification performance and informs the user. The disadvantages are problems with both statistical and computational efficiency. Aggregation loses information in the data, which increases the variance of classifier estimates and causes problems with both type 1 and type 2 errors in feature selection [11]. Searching a large space of potential features presents considerable computational challenges. For an example, generating 100,000 features on the standard CiteSeer dataset, can take several CPU days [20, Ch.16.1.2]). Compared to feature generation, the score aggregation approaches we describe in this paper provide a simple and fast baseline for link-based classification. At the end of the paper we describe possibilities for combining score-based aggregation with feature generation.

Sports Statistics. The problem of predicting the results of sports matches has received considerable attention for different sports. For an overview please see [22]. We do not claim that the methods in this paper are competitive for predicting the match results. We use the NHL data as a real-world dataset in an interesting domain with interpretable features for comparing aggregating features vs. aggregating predictions.

3 Notation and Data Format

We introduce notation to discuss relational features and data and to support theoretical analysis. We follow the functor-based notation for combining statistical and relational concepts due to Poole [19].

Functor Features. A **population** is a set of individuals, corresponding to a domain or type in logic. A **feature** is of the form $f(t_1, \ldots, t_k)$ where f is a functor (either a function symbol or a predicate symbol) and each t_i is a first-order variable or a constant. Each feature has a set of values (constants) called the **domain** of the feature. A feature whose range are the truth values $\{T, F\}$ is a **predicate**. Predicates are usually written with uppercase Roman letters, other feature with lowercase letters. A **grounding** replaces each 1st-order variable in the feature by a constant; the result is a ground feature. A grounding may be applied simultaneously to a set of features. One of the features is the class or **target** feature. A grounding of the target feature is a **target instance**.

Examples. In our datasets the basic populations are teams, players, matches, with corresponding first-order variables T, P, M. Examples of features include the following.

- $\mathit{result}(T, M)$ denotes the result of a team in a match (win or lose). This is the target feature.
- The ground feature $\mathit{result}(\mathit{Canucks}, \mathit{1})$ denotes the result of the Canucks in match 1. This is a target instance.
- $\mathit{PlaysFor}(P, T, M)$ is a predicate that is true if player P plays for team T in match M.

- $goals(T, P, M)$ is the number of goals scored by a player of a given team in a match.
- +/-(P, M) is the +/- score of a player in a match. This is a common measure of the player's performance; for precise definition see [22].

Aggregation. Given a feature f, an aggregate function *agg* applies to one of the argument variables of f. We use the subscript notation agg_X to indicate that variable X is the object of aggregation [20]. The result is a feature with one less argument. Examples include the following.

- $goals(T, M) \equiv \sum_P goals(T, P, M)$ is the number of goals scored by a team in a match.
- $past\_goals(P) \equiv \sum_{M \text{ in past season}} goals(T, P, M))$ denotes the sum of a player's goals in the past season.

Relational Data Tables. Relational data can be visualized in terms of the **groundings data table**. The data table has one column for each feature. It has one row for each simultaneous grounding of all functor features where the instances of the nonclass features are in the neighborhood of the instance of the target feature. Thus if the target functor feature is instantiated with ground instance t, the data table contains a row listing the attributes of each neighbor n of t. Tables 1 and 2 (propositionalized) show an example of groundings data tables. As the examples illustrate, aggregation increases the number of features (columns) and decreases the number of data points (rows).

Table 1. Data Table for NHL. Features include last season aggregates and target match statistics for players.

result(T,M)	MatchId M	TeamId T	PlayerId P	past_goals(P)	goals(T,P,M)
Loss	2010020023	Canucks	D. Hamhuis	5	0
Loss	2010020023	Canucks	D. Sedin	34	0
Loss	2010020023	Canucks	H.Sedin	32	0
...	...	...	(Player #18)	...	...
Win	2010020033	Canucks	D. Hamhuis	5	0
Win	2010020033	Canucks	C. Ehrhoff	17	0
Win	2010020033	Canucks	H. Sedin	32	0

4 Score Aggregation and Random Selection Classification

For simplicity, we discuss score aggregation for a single relationship, which defines a neighborhood for each grounding of the target feature. Our discussion applies equally to classification scores obtained with different types of neighbourhoods. Score aggregation can be visualized in terms of the **groundings data table**, or data table for short.

Table 2. Data Table for NHL. Features are team aggregates of last season player aggregates and aggregates of player target match statistics.

result(T,M)	MatchId M	TeamId T	Sum_past_goals(T)	Sum_goals(T,M)
Loss	2010020023	Canucks	252	1
Win	2010020033	Canucks	259	2

Suppose that we have trained a classifier model $\mathcal{M}$ that returns a classification score for a given target label y and feature vector $\boldsymbol{x}$. We write $score_{\mathcal{M}}(y; \boldsymbol{x})$. We can apply this classifier to each row in the groundings data table to derive a classification score from the features of each neighbour of a given target instance t. Given a list of classification scores, one for each row in which the target instance appears in the data table, we can apply a standard aggregation function to obtain an overall classification score. We also use the noisy-or rule for combining probabilities [12].

Random Selection Semantics We can also apply the classic random selection semantics to derive a score aggregation method in a principled way [10, 4]. Consider a probabilistic classifier score such as

$$0.7 = P_{\mathcal{M}}(result(T, M) = loss; past\_goals(P) = 10, goals(T, P, M) = 2)$$

On the random selection semantics, the meaning of this statement is "if we were to randomly select a team, match, and player, there is a 0.7 probability that the class label is loss." With this interpretation, for a fixed target instance $T = t$, the **random selection classification score** is the expected or *average* score over all the neighbours of the target instance. In terms of the data table, it is the average of the scores in the rows in which the target instance appears. Therefore *the random selection semantics leads to using the arithmetic mean for score aggregation.*

We can also use the random selection semantics for learning, not only prediction. A common framework for learning a classifier is to tune the classifier's parameters to minimize a loss function. For example, for probabilisic classifiers, a common loss function for a single data vector with class label y and features $\boldsymbol{x}$ is the log-likelihood loss $\ell_{\mathcal{M}}(y; \boldsymbol{x}) = -\ln P_{\mathcal{M}}(y; \boldsymbol{x})$. In terms of the data table, for a given classifier there is a loss in each row just as there is a score. Therefore the classifier's **random selection loss** on a single target instance is the average loss over the target's neighbours. The random selection loss on the entire dataset is the sum of the losses for each single target instance.

In comparison, given a data matrix as input, an optimization algorithm would assess the loss for each row, then sum the losses for each row. Without adjusting for neighbourhood sizes the summation loss is biased towards target instances with larger relational neighbourhoods. These observations have statistical and computational significance.

(1) One of the difficulties with applying standard loss minimization to relational data is that relational datapoints are not i.i.d., but the standard summation of classifier loss over data points assume data point independence. The random selection concept can be applied to define, in a principled way, a loss for an entire relational data table, based on a loss function for a single row.

(2) Adjusting the loss function for the size of different neighbourhoods requires only a small adjustment in the design of a standard classifier algorithm. If all target instances have the same number of neighbours, it is not necessary, and we can apply a standard classifier algorithm to the groundings data table.

Next we present empirical evaluation of feature and score aggregation methods.

5 Evaluation

Each method is evaluated by its ability to correctly classify the result based on the features supplied. For feature aggregation, this involves aggregating the supplied features and using these aggregated features to train a SVM or Logistic Ridge Regression model and return a result (win or loss) classification for each team per match.

For score aggregation, all features are first used to train a SVM or Logistic Ridge Regression model to learn a score for each row of features for a player in a given match. These player scores are then grouped for each team in a match, and the scores are aggregated to generate a team score and result classification.

5.1 Hardware and Software

The learning algorithms were executed with 64-bit Windows 7 with 12GB RAM and an Intel Core i7 2670QM CPU 2.2GHz processor. For Logistic Ridge Regression we used the L1General Matlab code [1]. For Support Vector Machine training we used the LibSVM software version 3.17.

5.2 Datasets

We prepared data tables from NHL and Premier League data. The target feature is $\mathit{result}(\mathit{Team}, \mathit{Match})$. A positive classification means that the team is predicted to lose the match. In the NHL, no ties are possible. In the Premier League dataset, we removed 90 tied matches from 380 total matches to obtain a comparable binary classification problem.

NHL data tables We used the webcrawler Selenium [3] to download player game statistics (Box Scores) from `http://www.nhl.com/gamecenter/` for the seasons 2009–2013. The box scores summarize player statistics for each match, a total of 13 continuous-valued features. We only consider skaters, no goalies. We refer to these as **match statistics**. The match features include goals, assists, plus-minus, penalty minutes, and total time on ice. For each player, we sum his match statistics over all NHL games in the previous season, to obtain a total of 13 season totals. In addition, we add 5 other season statistics: number of games played, game winning goals, powerplay goals, shorthanded goals, and shot percentage. We refer to the resulting 18 features as **last-season statistics**. From this data we prepared the following 4 data tables, one for each choice of (aggregate/not aggregate) and (last-season/all features).

Last Season A row in this table corresponds to a match, one of the teams involved in the match, and one player who played for that team in the match. Each NHL team dresses exactly 18 skaters per match, so for a given match, the data table contains 2 × 18 = 36 rows. The 19 columns represent the result of the match, and the 18 last-season statistics of the player.

All Features This data table contains the last season features and the match statistics. Table 1 illustrates this design.

Last Season Aggregate We apply the 6 aggregate functions max, min, sum, midpoint, average (arithmetic mean), geometric mean to each last-season feature.

All Features Aggregate Same as All Features Aggregate but with aggregates of match statistics as well. Table 2 illustrates this design.

Since each NHL team dresses exactly 18 skaters per match, the size of the relational neighbourhood for each (team, match) target instance is constantly 18. Thus the random selection loss for score aggregation can be optimized by applying a standard learning algorithm to the data table as is. For the propositionalization, it is known that aggregation introduces statistical bias when the sizes of relational neighbourhoods differ [7], so this data set is favourable for propositionalization. Table 3 provides totals for the NHL data tables.

Table 3. Number of data points (rows) and features (columns) in the NHL data tables.

Data Table	Last Season	All Features	Last Season Aggregate	All Features Aggregate
Number Rows	44,280	44,280	2,460	2,460
Number Columns	19	32	109	187

Premier League Data The data tables are derived from the Opta data, released by Manchester City [2]. There are 199 match statistics. As the Opta dataset provides only data for the 2011-2012 season, we used the current match statistics only, for a total of 2 data tables, as shown in Table 4.

5.3 Methods Compared

As base classifiers, we use SVM and logistic ridge regression. For SVM, we report results for 4 kernels. Parameters of the classifiers were set by a grid search that

Table 4. Number of data points (rows) and features (columns) in the Premier League data tables.

Data Table	Match Features	Match Features Aggregate
Number Rows	4,207	308
Number Columns	200	1,195

evaluated a parameter setting by cross-validation on the training set. We report the results for the best parameter setting found by the grid search.

For feature aggregation, we report results for pairs (Classifier x Dataset), for a total of $5 \times 2 = 10$ measurements for the NHL data and 5×1 for the Premier League data. For score aggregation, we report results for triples (Classifer x Dataset x aggregate operator), for a total of $5 \times 2 \times 4 = 40$ measurements for the NHL data and $5 \times 1 \times 4 = 20$ for the Premier League data. Player scores are aggregated using average, maximum, minimum, and noisy-or.

5.4 Performance Measures

Our basic metrics are **classification accuracy** (percentage of correctly classified target instances) and **F-measure**, the harmonic mean of precision and recall. We train the classifiers on a training set of target instances and test on the remaining target instances. For the NHL, the training set is derived from the entire 2010-2011 NHL season, and the test set comprises the matches from the 2011-2012 and 2012-2013 seasons. For the Premier League, the training set comprises the first half of the 2011-2012 season, and the test set comprises the second half.

5.5 Results

We first compare different methods for score aggregation, then for feature aggregation, finally both together.

Score Aggregation. Table 5 shows the accuracies for each combination of (trained base classifier, score aggregator), for both feature sets for the NHL data. The F-measures showed the same trends. Logistic Ridge Regression achieved high results with both the average and maximum operator, for both probabilities and log probabilities. SVMs performed poorly overall, with the exceptions of using the linear or quadratic kernel with the average score aggregator. Logistic regression is superior to SVMs when used with score aggregation on these datasets.

Table 6 shows the results for the premier league data. Here the average operator performs clearly the best, especially with logistic regression probabilities. For other operators, SVMs do slightly better than logistic regression, but poorly overall.

In sum, logistic regression appears more robust on our datasets than SVM, and the average aggregator appears more robust than other aggregators.

Table 5. NHL Results. Logistic Regression is used both with the predicted probability (Prob) and the log-probability (Log-Prob). The random selection classifier uses Average + Log-Prob. Italics indicate the best SVM classifier.

Last Season	Score Aggregator			
	Average	Maximum	Minimum	Noisy-Or
SVM - Linear	*53.59%*	50.57%	**50.45%**	50.57%
SVM - Quadratic	50.89%	50.00%	50.00%	**50.69%**
SVM - Sigmoid	50.00%	50.00%	50.00%	50.00%
SVM - Gaussian	50.00%	50.00%	50.00%	50.00%
LR - Prob	**54.43%**	**54.43%**	50.42%	50.00%
LR - Log-Prob	54.39%	**54.43%**	50.42%	N/A

All Features	Score Aggregator			
	Average	Maximum	Minimum	Noisy-Or
SVM - Linear	73.46%	50.73%	**57.51%**	**52.82%**
SVM - Quadratic	*80.91%*	53.78%	53.92%	51.87%
SVM - Sigmoid	50.00%	50.00%	50.00%	50.00%
SVM - Gaussian	50.00%	50.00%	50.00%	50.00%
LR - Prob	**87.26%**	**87.26%**	54.89%	51.67%
LR - Log-Prob	86.60%	**87.26%**	54.89%	N/A

Table 6. PLG Target Match

	Score Aggregator			
	Average	Maximum	Minimum	Noisy-Or
SVM - Linear	*80.15%*	**54.78%**	51.10%	54.78%
SVM - Quadratic	76.84%	53.68%	51.47%	**53.68%**
SVM - Sigmoid	53.68%	50.00%	50.00%	50.00%
SVM - Gaussian	68.75%	50.37%	**55.88%**	50.37%
LR - Prob	**81.62%**	52.57%	52.21%	52.57%
LR - Log-Prob	77.21%	52.57%	52.21%	N/A

Feature Aggregation Table 7 presents the results for feature aggregation methods, on all three datasets. Examination of regression weights showed minimum and sum of match plus-minus to be a strong predictor for NHL results. Logistic regression with feature aggregation on the NHL All Features dataset showed a 25.67% improvement over all SVM models.

Score Aggregation vs. Feature Aggregation. Table 8 compares the best feature aggregation method with the best score aggregation method. On the NHL datasets, feature Aggregation performs at least as well as Score Aggregation, although the difference is not statistically significant. Feature aggregation clearly performs better on the Premier League datasets.

In sum, comparing score aggregators, the average suggested by the random selection semantics appears to be the most robust aggregator, providing competitive performance in a variety of settings. Minimum and noisy-or operators performed poorly on all datasets. Comparing base classifiers, logistic regression

Table 7. Feature Aggregation Results

	Data Table		
Classifier	NHL Last Season	NHL All Features	PLG Match Features
SVM - Linear	*53.43%*	60.40%	**90.44%**
SVM - Quadratic	52.66%	61.74%	89.71%
SVM - Sigmoid	50.00%	50.00%	50.00%
SVM - Gaussian	50.00%	52.52%	54.78%
Logistic Regression	**53.94%**	**87.41%**	**90.44%**

Table 8. Comparison of Classification Accuracy and F-Measure for Feature and Score Aggregation

	Feature Aggregation		Score Aggregation	
Dataset	Classification Accuracy	F-Measure	Classification Accuracy	F-Measure
NHL Last Season Stats	53.94%	0.5368	54.53%	0.5593
NHL All Features	87.41%	0.8753	87.26%	0.8709
PLG Match Features	90.44%	0.9044	81.62%	0.8120

appears more reliable than SVMs, which can perform very poorly with the wrong kernel.

A hybrid approach that combines score aggregation with feature aggregation could address the weaknesses of both approaches. For example good features could be found learning a model based on feature aggregation. Adding good aggregation features to non-aggregated features (e.g., player statistics) in score aggregation could then improve classification accuracy. Conversely, a score-aggregation classifier can be used as a strong baseline for pruning noninformative aggregate features, to reduce the expensive search through the aggregation space.

6 Conclusion

We considered link-based classification with continuous features of linked entities, on real-world sports data sets. Two basic approaches are aggregating features vs. aggregating classifier scores. Empirical results indicate that averaging is the most robust operation for classifier scores. Averaging is consistent with the classic random selection semantics for probabilistic logic. A promising avenue for future work is to combine aggregate features with score aggregation.

References

[1] The l1general matlab package. URL = `http://www.di.ens.fr/~mschmidt/Software/L1General.html`.

[2] Mcfc analytics. URL = `http://www.mcfc.co.uk/Home/The%20Club/MCFC%20Analytics`.

[3] Selenium. URL = `http://www.seleniumhq.org/`.

[4] F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities*. MIT Press, Cambridge, MA, USA, 1990.

[5] B. Bina, O. Schulte, B. Crawford, Z. Qian, and Y. Xiong. Simple decision forests for multi-relational classification. *Decision Support Systems*, 54(0):1269C1279, 2013.

[6] H. Chen, H. Liu, J. Han, and X. Yin. Exploring optimization of semantic relationship graph for multi-relational Bayesian classification. *Decision Support Systems*, 48(1):112–121, 2009.

[7] J. N. David Jensen. Linkage and autocorrelation cause feature selection bias in relational learning (2002). In *ICML*, 2002.

[8] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.

[9] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[10] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46(3):311–350, 1990.

[11] D. Jensen and J. Neville. Data mining in social networks. In *In National Academy of Sciences workshop on Dynamic Social Network Modeling and Analysis*, 2002.

[12] K. Kersting and L. de Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [9], chapter 10, pages 291–318.

[13] S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In *Relational Data Mining*, pages 262–286. Springer, 2000.

[14] M.-A. Krogel and S. Wrobel. Feature selection for propositionalization. In *Discovery Science*, pages 430–434, London, UK, 2002. Springer-Verlag.

[15] O. Kuzelka and F. Zelezný. Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning*, 83(2):163–192, 2011.

[16] Q. Lu and L. Getoor. Link-based classification. In *ICML*, pages 496–503. AAAI Press, 2003.

[17] S. Natarajan, P. Tadepalli, T. G. Dietterich, and A. Fern. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artifical Intelligence*, 54(1-3):223–256, 2008.

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[19] D. Poole. First-order probabilistic inference. In *IJCAI*, pages 985–991, 2003.

[20] A. Popescul and L. Ungar. Feature generation and selection in multi-relational learning. In *Introduction to Statistical Relational Learning* [9], chapter 16, pages 453–476.

[21] O. Schulte. A tractable pseudo-likelihood function for Bayes nets applied to relational data. In *SIAM SDM*, pages 462–473, 2011.

[22] R. P. Schumaker, O. K. Solieman, and H. Chen. *Research in Sports Statistics*, volume 26 of *Integrated Series in Information Systems*. Springer US, 2010.

[23] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning* [9], chapter 4, pages 93–127.

[24] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, pages 485–492. Morgan Kaufmann Publishers Inc., 2002.

[25] A. Van Assche, C. Vens, H. Blockeel, and S. Dvzeroski. First order random forests: Learning relational classifiers with complex aggregates. *Machine Learning*, 64(1):149–182, 2006.