Propositionalization for Unsupervised Outlier Detection in Multi-Relational Data

Fatemeh Riahi and **Oliver Schulte**
Simon Fraser University,
sriahi@sfu.ca, oschulte@cs.sfu.ca

Abstract

We develop a novel propositionalization approach to unsupervised outlier detection for multi-relational data. Propositionalization summarizes the information from multi-relational data, that are typically stored in multiple tables, in a single data table. The columns in the data table represent conjunctive relational features that are learned from the data. An advantage of propositionalization is that it facilitates applying the many previous outlier detection methods that were designed for single-table data. We show that conjunctive features for outlier detection can be learned from data using statistical-relational methods. Specifically, we apply Markov Logic Network structure learning. Compared to baseline propositionalization methods, Markov Logic propositionalization produces the most compact data tables, whose attributes capture the most complex multi-relational correlations. We apply three representative outlier detection methods (LOF, $KNNOutlier$, $OutRank$) to the data tables constructed by propositionalization. For each outlier detection method, Markov Logic propositionalization provided the best average accuracy over all datasets.

Introduction

Many outlier detection methods have been developed for data single-table data in attribute-value format (Aggarwal 2013). This paper addresses outlier detection for multi-relational data. Given the prevalence of relational data in organizations, outlier analysis for such data is an important problem in practice. However, applying standard outlier methods designed for single data tables runs into an impedance mismatch, since multi-relational data are represented in multiple interrelated tables. Single-table tools can be leveraged for multiple data tables via a pipeline data preprocessing approach: first, convert the multi-relational data to a single attribute-value table, then apply the data analysis tool. Since the attribute value representation corresponds to propositional logic, the conversion process is called *propositionalization* (Lippi et al. 2011). While propositionalization for classification has been extensively explored, outlier detection is a new deployment context for propositionalization.

Approach. The input to a propositionalizer is a relational database, and the output is a data matrix in attribute-value format. Each column (attribute) of the data matrix is associated with a conjunctive logical formula called the query. The query formula is a *template* that can be instantiated multiple times for a single individual. The value of the attribute for an individual is determined by a function that aggregates the multiple instantiations to derive a real number. We use Markov Logic Network (MLN) structure learning to *learn a set of informative formulas from an input relational database.* This is a novel application of MLN learning. Our approach can be summarized by the equation

Markov Logic Network Structure = Set of Formulas =
Set of Attributes in Data Table.

Motivation for Markov Logic Networks. Constructing a generative model is one of the major approaches to unsupervised outlier detection (Aggarwal 2013). Intuitively, the generative model represents normal behavior in the population. MLNs are one of the main generative model classes for relational data (Domingos and Lowd 2009). The formulas in the MLN indicate which relations are normally associated, and which are normally independent of each other.

Evaluation. We use three synthetic and two real-world datasets, from the UK Premier Soccer League and the Internet Movie Database (IMDb). Our baseline is an n-gram approach: Enumerate all conjunctive formulas up to length 1 or 2 as attributes for propositionalization (Perovsek et al. 2013). Markov Logic propositionalization shows three advantages. 1) Many *fewer formulas/attributes*, leading to much smaller data tables for outlier analysis. 2) *Longer formulas* that capture more complex relational associations. 3) *Accuracy:* For a given outlier analysis method, the average Area-Under-Curve (AUC) score over all datasets is best for MLN propositionalization.

Contributions. Our contributions may be summarized as follows.

1. A novel task for propositionalization: supporting outlier detection.
2. A novel application of Markov Logic Network structure learning to perform this task.

Paper Organization. We review related work. Then we introduce notation for describing relational data. The main section describes how Markov Logic Networks can be applied to learn logical formulas. We evaluate Markov Logic propositionalization against baseline methods.

Related Work

Propositionalization has been explored extensively for classification tasks (Kramer, Lavrac, and Flach 2000; Perovsek et al. 2013; Kuvzelka and Zelezny 2008). We are not aware of previous work on propositionalization for *outlier detection.* The **wordification** framework (Perovsek et al. 2013) introduces an analogy between the grounding counts of a formula and the term frequency in a document. Lavrac *et al.* apply this analogy for propositionalization for classifiers. The n-gram methods in this paper apply the analogy for outlier detection.

In previous applications of single-table outlier analysis methods to structured data, the data were manually propositionalized, by aggregating information about single attributes (Breunig et al. 2000). For example, Breunig *et al.* counted the total number of goals scored by a player in a season as a attribute for outlier analysis by LOF. This is equivalent to the unigram term frequency method we evaluate. Manual propositionalization is limited because capturing interactions between features is difficult.

Relational Outlier Detection has previously been based on discovering *rules* that represent the presence of anomalous associations for an individual or the absence of normal associations (Novak, Lavravc, and Webb 2009; Maervoet et al. 2012). Related work includes exception mining, which looks for associations that characterize unusual cases, subgroup mining, which looks for associations characterizing important subgroups (Atzmueller 2015), and contrast space mining, which looks for differences between classes. Novak *et al.* show that these tasks can be unified as instantiations of a general rule search framework. **Inductive Logic Programming** has been used for learning relational outlier detection rules. One approach views an example as anomalous if it is not covered by a learned set of rules (Angiuli and Palopoli 2007). Another measures the difference in generality between a rule set learned with the anomalous examples, and a rule set learned without (Angiuli and Palopoli 2007). None of the rule search methods use propositionalization.

Background and Notation

We use notation and terminology from previous work (Chiang and Poole 2012; Lippi et al. 2011; Domingos and Lowd 2009). While we do not introduce any new terminology, we combine concepts from different areas, such as propositionalization and log-linear models.

Logical Concepts

We adopt a term-based notation for logical concepts (Poole 2003; Chiang and Poole 2012). Constants such as *rooney*, 123 are used to represent individuals. A **population** is a set of individuals. A functor is a function or predicate symbol, denoting a function that is applied to individuals. Each functor has a set of possible values (constants) called the **domain** of the functor. The domain of a **predicate** is $\{T, F\}$. Predicates are usually written with uppercase Roman letters, other terms with lowercase letters. A predicate of arity at least two is a **relationship** functor. Relationship functors specify which objects are linked. Other functors represent **features** or **attributes** of an object or a tuple of objects (i.e., of a relationship). A **term** is of the form $f(\sigma_1, \ldots, \sigma_k)$ where f is a functor and each σ_i is a first-order variable or a constant. An *atomic assignment* specifies the value of a term e.g. $f(\sigma_1, \ldots, \sigma_k) = v$ where v is in the domain of functor f. Connecting assignments using only $\wedge$ forms a *conjunctive formula* or *conjunction*. In this paper we use conjunctive formulas only. A term/literal/formula is **ground** if it contains no first-order variables; otherwise it is a first-order term/literal/formula.

A relational database $\mathcal{D}$ specifies the values of all ground terms, and hence whether a ground literal is true or not. A ground conjunction is true in a database if all of its conjuncts are true. A **grounding** is a set $\{X_1 \backslash x_1, \ldots, X_k \backslash x_k\}$ where X_i are distinct logical variables and x_i are constants. When applied to a formula, each occurrence of X_i is replaced with x_i. The **count** of groundings that satisfy (make true) formula ϕ with respect to a database $\mathcal{D}$ is denoted as $\#_{\mathcal{D}}(\phi)$.

Markov Logic Networks A Markov Logic Network (MLN) (Domingos and Lowd 2009) is a set $\{(\phi_1, w_1), \ldots, (\phi_m, w_m)\}$ where ϕ_i is a formula, and each w_i is a real number called the weight of ϕ_i. The MLN semantics views a formula with logical variables as a feature template that is instantiated by ground formulas. The number m of formulas is independent of the size of the instantiated MLN. The log-linear likelihood of a possible world/database is proportional to the weighted sum, over all formulas, of the grounding count of each formula in the given database:

$$P(D) \propto \exp(\sum_{i=1}^{m} w_i \cdot \#_{\mathcal{D}}(\phi_i)) \tag{1}$$

This semantics defines a joint distribution over descriptive attributes of entities, links between entities, and attributes of those links. Domingos and Lowd discuss the representational power of this semantics.

Examples Figure 1(a) shows an example database. The ground literal

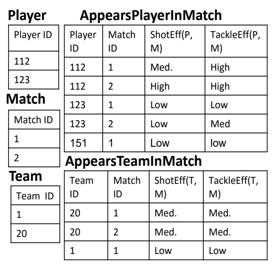
$$(\mathit{ShotEff}(P, M) = \mathit{Low})\{P\backslash \mathit{123}, M\backslash \mathit{1}\}$$
$$= (\mathit{ShotEff}(\mathit{123}, \mathit{1}) = \mathit{Low})$$

evaluates as true in this database. For the grounding count we have

$$\#_{\mathcal{D}}(\mathit{ShotEff}(P, M) = \mathit{Low})\{P\backslash \mathit{123}\}) = \mathit{2}.$$

Propositionalization via Markov Logic Networks

Figure 1(b) provides an overview of our propositionalization system. We follow the description of propositionalization by (Lippi et al. 2011). The output of the propositionalizer as a **pseudo-iid data view**, because the data matrix is



Player

Player ID
112
123

Match

Match ID
1
2

Team

Team ID
1
20

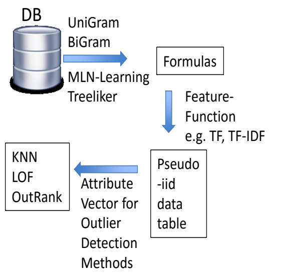
AppearsPlayerInMatch

Player ID	Match ID	ShotEff(P, M)	TackleEff(P, M)
112	1	Med.	High
112	2	High	High
123	1	Low	Low
123	2	Low	Med
151	1	Low	low

AppearsTeamInMatch

Team ID	Match ID	ShotEff(T, M)	TackleEff(T, M)
20	1	Med.	Med.
20	2	Med.	Med.
1	1	Low	Low

(a) Example database (DB)



(b) Propositionalization Pipeline

Figure 1: Relational Data and System Flow

Algorithm 1 Markov Logic Network Propositionalization

```
Input: An MLN $\{(\phi_1, w_1), \ldots, (\phi_m, w_m)\}$; Database $\mathcal{D}$; Example logical variable $E$.
Output: A data matrix $D$. (Pseudo-iid data view.)
Calls: Feature Function $F$. $F(a, \phi, \mathcal{D})$ returns a number.
1: For each individual $a_1, \ldots, a_n$ in the domain of the example variable $E$, add a row to the data matrix $D$.
2: For each formula $\phi_1, \ldots, \phi_m$ in the MLN that contains the example variable, add a column to the data matrix $D$.
3: for all individuals $a_i$ and formulas $\phi_j$ do
4:     $D_{ij} := F(a_i, \phi_j, \mathcal{D})$.
5: end for
6: Return $D$.
```

processed by methods designed for i.i.d. data, even though the relational structure induces dependencies between rows in the data matrix. Each column (attribute) of the pseudo-iid data view is associated with a conjunctive formula called the query. The query contains a logical **example variable** E. For instance, if we are interested in detecting anomalous players, $Player$ is an appropriate example variable. For each example individual, the value of an attribute is determined by a function that aggregates the multiple instantiations of the attribute query for the individual, to derive a real number. In the terminology of log-linear models like MLNs, such functions are called **feature functions** (Domingos and Lowd 2009). Table 1 illustrates feature functions. In sum, pseudo-iid data views are constructed as follows.

Formula + Feature Function = Attribute Values

Algorithm 1 describes how this propositionalization schema can be applied with Markov Logic Networks.

Table 1: A formula with different feature functions. For definitions please see text. Numbers were computed from the toy database shown in Figure1(a). Each column shows a combination of (formula, feature function) that defines an attribute. Our system learns multiple such formulas for defining attributes from the data.

Formula →	$Shot\_Eff(P, M) = high \wedge Tackle\_Eff(P, M) = high$	
Feature Function → Player ↓	*TF*	*TF-IDF*
112	1	$1 \times \log_2(3/1) \approx 1.58$
123	0	0

Table 2: Generating pseudo-iid data views using Feature Functions and Formulas

Feature Function → Formula ↓	*TF*	*TF-IDF*
Unigram	Unigram-TF	Unigram-IDF
Bigram	Bigram-TF	Bigram-IDF
MLN	MLN-TF	MLN-IDF

Feature Functions: Wordification and n-grams

A number of feature functions have been established for text data. The recent wordification analogy between relational and text data allows us to transfer concepts from one domain to the other (Perovsek et al. 2013). The analogy is as follows. A document corresponds to an example individual. A word in a document corresponds to a literal. An n-gram in a document (i.e., a sequence of n words) corresponds to a conjunction of n literals. The term frequency (TF) of an n-gram in a document corresponds to the grounding count.

In NLP/IR research, a widely used feature function is term frequency/inverse document frequency ($TF - IDF$), which down-weights terms that are frequent across documents. For a given w in document d from corpus D, the $TF - IDF$ measure is defined as follows:

$$TF - IDF(w, d) = TF(w, d) \times \log_2 \frac{|D|}{d \in D : w \in d}$$

Learning Attributes/Formulas for Unsupervised Propositionalization

Wordification suggests an approach to generating formulas: just as text mining often constructs all n-grams up to a feasible bound n, we can *enumerate all conjunctions of* n *literals* containing the example variable. In our datasets, this was computationally feasible for $n < 3$.

We apply MLN structure learning to *learn* a compact set of relevant formulas. In this paper we employ the previously existing Learn-and-Join (LAJ) algorithm. This is a state-of-the-art MLN structure learning algorithm, especially well-suited for datasets with many descriptive attributes such as those in our empirical evaluation (Schulte and Khosravi 2012). The Learn-and-Join algorithm employs an iterative deepening strategy that searches for correlations among increasingly longer chains of relationships. Correlations discovered for shorter chains are propagated to longer chains. In sum, we use the following methods for generating attributes/formulas in a pseudo-iid data view.

MLN Learn a Markov Logic Network structure for the input database, then use the learned formulas.

Unigram All single literals.

Bigram All conjunctions of two literals that share at least one first-order variable.

Experimental Design: Methods Used

The pairs (Formula Generation, Feature Function) define six propositionalization methods; see Table 2. For Unigram and MLN attribute generation, the IDF feature function produced worse results than TF on all datasets, so we omit results for these two methods. Understanding the negative impact of IDF downweighting on outlier detection is a valuable topic for future work. Instantiation counts for term frequencies and inverse document frequencies were computed using MySQL Server version 5.5.34 run with 8GB of RAM and a single core processor of 2.2GHz. For the Learn-and-Join structure learning algorithm, we used the implementation due to its creators (Qian and Schulte 2015). Our code is available on-line.[1]

Outlier Analysis Methods. The output of all 4 propositionalization methods is provided to the following 3 standard outlier analysis techniques for data matrices, which represent three fundamental approaches to outlier detection. Our design space therefore contains 12 complete relational outlier detection algorithms.

LOF is a standard density-based method (Breunig et al. 2000). LOF compares the density of area around an object to the densities of the areas of the surrounding objects.

$KNNOutlier$ is a well-known distanced-based outlier ranking method that assigns a score to each data point on the basis of distance of the point from its k^{th} nearest neighbor (D^k) and declare the top n points as outliers (Ramaswamy, Rastogi, and Shim 2000).

$OutRank$ employs subspace analysis to measure outlierness. It compares clusters in different subspaces to derive an outlier score for each object.

System Details. Both LOF and $KNNOutlier$ require specifying the value of a k parameter. Following the recommendation of the LOF creators (Breunig et al. 2000), we employed the three k-values 10,15,20. Our experiments report the best results. The $OutRank$ research of (Muller et al. 2012) suggest using *DISH* or *PRO-CLUS* as clustering subroutines; our experiments applied *DISH* Outrank requires three parameters to be specified, α, ϵ and μ. For these parameters we tested different values in the suggested range and the experiments reports the best results. We used the available implementation of all three data matrix methods from the state of the art data mining software *ELKI* (Achtert et al. 2013).

Experimental Design: Datasets Used

We give a brief description of our datasets. For complete information about their provenance and format, see (Riahi and Schulte 2015).

Real-World Datasets

We used the soccer `PremierLeague` database, and the movie database `imdb_MovieLens`, both available from the Prague Relational Learning Repository (Motl and Schulte 2015).

Soccer Data The database lists all the ball actions within each game by each player, for the 2011-2012 season. For each player in a match, our data set contains eleven attributes of a player in a match, like $TimePlayed(P, M)$. There are two relationships, $Appears\_Player(P, M)$, $Appears\_Team(T, M)$.

[1]ftp://ftp.fas.sfu.ca/pub/cs/oschulte/CodesAndDatasets/.

Table 3: Outlier vs. normal Objects in Real Datasets.

Normal	#Normal	Outlier	#Outlier
Striker	153	Goalie	22
Midfielder	155	Striker	34
Drama	197	Comedy	47

IMDb Data The Internet Movie Database (IMDb) is an on-line database of information related to films, television programs, and video games. The database contains seven tables: one each for the four populations $Users, Movies, Actors, Directors$ and one for the three relationships $Rated(User, Movie)$, $Directs(Director, Movie)$, and $ActsIn(Actor, Movie)$.

In real-world data, there is no ground truth about which objects are outliers. We employ a one-class design (Riahi and Schulte 2015): learn a model for the class distribution, with data from that class only. Following (Gao et al. 2010; Aggarwal 2013) we rank all individuals from the normal class (e.g. Striker class) together with all objects from a contrast class (e.g. Goalie class) treated as outliers, to test whether an outlier score recognizes objects from the contrast class as outliers. Table 3 shows the normal and contrast classes for three different datasets. In-class outliers are possible, e.g. unusual strikers are members of the striker class.

Synthetic Datasets

We generated three synthetic datasets with normal and outlier players. Each player participates in 38 matches, similar to the real-world data. Each match assigns a value to only two features F_1 and F_2 for each player, according to the following distributions.

High Correlation Normal individual: strong feature association. Outlier: no association.

Low Correlation Normal individual: no feature association. Outlier: strong association.

Single features Feature associations are the same but marginal feature distributions are very different.

We used the $mlbench$ package in R to generate synthetic features in matches that exhibit these association patterns for 240 normal players and 40 outliers. The probabilistic associations are chosen to be so strong that outliers are evident (for complete details, please see (Riahi and Schulte 2015)).

Evaluation Results

Performance Metrics Used

We report several properties of the pseudo-iid data views produced by the different methods.

Dimensionality The number of attributes in the pseudo-iid data view.

Attribute Complexity The length of the conjunctions that define the attributes.

Table 5: OutRank running time (ms) given different attribute vectors for TF (IDF performance is very similar). Running times for other outlier analysis methods are essentially the same.

Dataset	$Unigram-TF$	$Bigram-TF$	$MLN-TF$
Drama vs. Comedy	945	855,714	389,765
MidFielders vs. Strikers	486	642,261	18,737
Strikers vs. Goalies	578	814,807	55,448

Outlier Analysis Run Time How long it takes each outlier method to rank outliers, given the pseudo-iid data view.

Attribute Construction Time How long it takes to build the pseudo-iid view from an input relational database.

Our performance accuracy score for outlier rankings is the area under curve (AUC) of the well-established receiver operating characteristic ROC curve (Aggarwal 2013). We also used precision (Gao et al. 2010) with similar results (not reported). To compute the AUC value, we used the *R* package *ROCR*. Given a set of outlier scores, one for each object, this package returns an AUC value. The summary of our findings is that MLN propositionalization shows the following advantages and disadvantages.

- For a fixed outlier detection method, competitive accuracy over all datasets (the best for LOF and $KNNOutlier$, tie with Bigram-idf for $OutRank$).
- Compact yet complex pseudo-iid data views: substantially fewer attributes (columns) than bigrams, yet average formula length 3.27 or greater.
- Faster outlier analysis due to this compactness.
- There is learning overhead for discovering relevant formulas, but it is small (e.g. 5 minutes for MLN learning vs. 1 minute for bigram construction).

Dimensionality of Pseudo-iid Data Views

Table 4 provides information about the formulas constructed by the different propositionalization methods, and the size of the resulting data table. The average formula length for MLNs is above 3 for the soccer data, for the IMDb data above 4. This shows that MLN structure learning finds more complex formulas beyond length 2. For the dimensionality of the resulting pseudo-iid views, there is a big increase from unigrams to bigrams (e.g. from 63 to 1825 for Strikers vs. Goalies). The dimensionality of MLN pseudo-iid data views lies between that of unigrams and bigrams, (e.g. 331 for Strikers vs. Goalies.) This shows that MLN structure learning can find complex longer formulas with a relatively small increase in the dimensionality of the resulting pseudo-iid data view, compared to bigrams. The trade-off is that learning a compact set of relevant formulas takes more time than enumerating all formulas up to a fixed length. However, the learning overhead is small (e.g. 5.24 min vs. 1.2 min for Strikers vs. Goalies).

Accuracy

Table 6 summarizes the performance of the propositionalization methods for each outlier detection algorithm. There is no single propositionalization method that always leads to the best accuracy for all three outlier analysis methods. MLN-TF propositionalization produces the best results on two datasets. It is always close to the best AUC score (never less than 0.1 AUC units away). The $\mu(AUC)$ column reports the average AUC score over different datasets. A propositionalization method "wins" on a dataset if its AUC is at least 0.01 greater than that of others. A "tie" for first place earns 0.5 points. The total number of points is shown in the Wins columns. MLN-TF comes out as the best method in terms of average AUC, for all outlier detection methods. MLN-TF propositionalization scores the most wins when applied with LOF or $KNNOutlier$, and a tie when applied with $OutRank$. Thus methods that tend to treat features independently, such as LOF and $KNNOutlier$, benefit from being provided complex attributes.

Table 6: A propositionalization method is scored 1 point if it produces the best accuracy on a dataset, and 0.5 points if it ties. The table shows the total number of wins and average of AUC over all datasets.

Propositionalization → Outlier Detection Method ↓	MLN-TF		Bigram-IDF		Unigram-TF		Treeliker	
	Wins	$\mu(AUC)$	Wins	$\mu(AUC)$	Wins	$\mu(AUC)$	Wins	$\mu(AUC)$
OutRank	**2.50**	**0.79**	**2.50**	0.70	1.00	0.64	0	NA
KNN	**3.50**	**0.78**	1.50	0.67	1.50	0.67	0	0.64
LOF	**4.00**	**0.63**	1.00	0.55	1.00	0.61	1	0.61

Table 7: AUC of Loglikelihood vs. MLN propositionalization

Dataset	Loglikelihood	MLN-TF-OutRank
Low Correlation	0.87	0.89
High Correlation	0.97	0.98
Single Feature	0.87	0.88
Drama vs. Comedy	0.23	0.68
Midfielder vs. Striker	0.43	0.70
Strikers vs Goalie	0.41	0.60

Other Baseline Methods

Log-likelihood Outlier Score

As of the submission deadline, the most recent relational outlier detection method that we know of was proposed by (Riahi, Schulte, and Li 2013). It is the only relational outlier algorithm for which we were able to find an implementation. The log-likelihood of an individual *database* indicates a potential outlier. The individual database is a dataview that contains the attributes and links of the potential outlier and related objects. (Individual databases are called interpretations in ILP research, see (Maervoet et al. 2012)). Using the notation of algorithm 1 with Equation (1), the relational log-likelihood score L for potential outlier a is a sum over formulas of (weight × term frequency):

$$L(a) \equiv \sum_{i=1}^{m} w_i \cdot \#_{\mathcal{D}}(\phi_i\{E/a\}).$$

Results. Table 7 reports results for MLN-TF-Outrank, which had the highest average AUC among the MLN methods. On the all datasets, MLN-TF-Outrank outperforms the log-likelihood score. On the real-world datasets, MLN -TF-Outrank's performance is much better.

Table 4:

Formula	MLN			Bigram			Unigram		
Dataset↓	μ(Formula Length)	Dimensionality	Construction Time(min)	μ(Formula Length)	Dimensionality	Construction Time(min)	μ(Formula Length)	Dimensionality	Construction Time(min)
Strikers vs. Goalies	3.55	331	5.24	2	1825	1.2	1	63	0.10
MidFielders vs. Strikers	3.27	198	4.92	2	1762	0.85	1	62	0.08
Drama vs. Comedy	4.20	930	10.80	2	1991	2.87	1	47	0.09

Conclusion

Many outlier analysis techniques have been developed for i.i.d. propositional data. To leverage these for finding outliers in multi-relational data, a pipeline propositionalization approach summarizes the information from multiple data tables in a single data table. The key is to find a set of relevant logical formulas that define attributes of potential outlier individuals. Rather than use a fixed a priori set of templates, *relevant formulas are learned from the data* using Markov Logic Network structure learning.

In an empirical comparison with the baseline wordification approach of enumerating all conjunctive formulas up to length 2, Markov Logic propositionalization showed several advantages: 1) The set of formulas learned was substantially smaller, leading to smaller data tables and faster outlier detection 2) The formulas learned were longer, representing more complex relational patterns. 3) For a fixed single-table outlier analysis method, the average detection accuracy was higher.

Future Work. Different approaches to learning query formulas include the following. 1) Different MLN structure learning algorithms. 2) Learning generative models other than MLNs. 3) Relational association rule learning (Deshape and Toivonen 2001), where a learned association rule is used as a query formula.

References

Achtert, E.; Kriegel, H.; Schubert, E.; and Zimek, A. 2013. Interactive data mining with 3d-parallel coordinate trees. In *Proceedings of the ACM SIGMOD*.

Aggarwal, C. 2013. *Outlier Analysis*. Springer New York.

Angiuli, F., and Palopoli, L. 2007. Outlier detection by logic programming. *ACM Computer Logic* 9.

Atzmueller, M. 2015. Subgroup Discovery - Advanced Review. *WIREs: Data Mining and Knowledge Discovery* 5:35–49.

Breunig, M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. Lof: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD*.

Chiang, M., and Poole, D. 2012. Reference classes and relational learning. *Journal of Approximation Reasoning* 53(3).

Deshape, L., and Toivonen, H. 2001. Discovery of relational association rules. In Dzeroski, S., and Lavrac, N., eds., *Relational Data Mining*. Springer, Berlin. chapter 8.

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

Gao, J.; Liang, F.; Fan, W.; Wang, C.; Sun, Y.; and Han, J. 2010. On community outliers and their detection in information networks. In *Proceedings of ACM SIGKDD*.

Kramer, S.; Lavrac, N.; and Flach, P. 2000. Propositionalization approaches to relational data mining. In *Relational Data Mining*. Springer.

Kuvzelka, O., and Zelezny, F. 2008. Hifi: Tractable propositionalization through hierarchical feature construction. In *Late Breaking Papers, ILP*.

Lippi, M.; Jaeger, M.; Frasconi, P.; and Passerini, A. 2011. Relational information gain. *Machine Learning* 83(2).

Maervoet, J.; Vens, C.; Vanden Berghe, G.; Blockeel, H.; and De Causmaecker, P. 2012. Outlier detection in relational data: A case study. *Expert Syst. Appl.*

Motl, J., and Schulte, O. 2015. The CTU prague relational learning repository. *CoRR* abs/1511.03086.

Muller, E.; Assent, I.; Iglesias, P.; Mulle, Y.; and Bohm, K. 2012. Outlier ranking via subspace analysis in multiple views of the data. In *IEEE ICDM*.

Novak, P. K.; Lavravc, N.; and Webb, G. I. 2009. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *The Journal of Machine Learning Research* 10:377–403.

Perovsek, M.; Vavpetic, A.; Cestnik, B.; and Lavrac, N. 2013. A wordification approach to relational data mining. In *Discovery Science*, Lecture Notes in Computer Science.

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

Qian, Z., and Schulte, O. 2015. The BayesBase system. `www.cs.sfu.ca/~oschulte/BayesBase/BayesBase.html`.

Ramaswamy, S.; Rastogi, R.; and Shim, K. 2000. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, volume 29, 427–438. ACM.

Riahi, F., and Schulte, O. 2015. Model-based outlier detection for object-relational data. In *IEEE Symposium Series on Computational Intelligence (SSCI*.

Riahi, F.; Schulte, O.; and Li, Q. 2013. Identifying important nodes in relational data. In *AAAI Late Breaking Paper*.

Schulte, O., and Khosravi, H. 2012. Learning graphical models for relational data via lattice search. *Journal of Machine Learning Research*.