k-RNN: k-Relational Nearest Neighbour Algorithm

Nuno A. Fonseca
Instituto de Biologia Molecular
e Celular (IBMC)
R. Campo Alegre, 823
4150-180 Porto, Portugal
nf@ibmc.up.pt

Vítor Santos Costa,
Ricardo Rocha
DCC-FCUP, University of
Porto
R. Campo Alegre, 1021/1055
4169-007 Porto, Portugal
{vsc,ricroc}@dcc.fc.up.pt

Rui Camacho
Faculty of Engineering &
LIAAD, University of Porto
R. Dr. Roberto Frias, s/n
4200-465 Porto, Portugal
rcamacho@fe.up.pt

ABSTRACT

The amount of data collected and stored in databases is growing considerably in almost all areas of human activity. In complex applications the data involves several relations and proposionalization is not a suitable approach. Multi-Relational Data Mining algorithms can analyze data from multiple relations, with no need to transform the data into a single table, but are computationally more expensive. In this paper a novel relational classification algorithm based on the k-nearest neighbour algorithm is presented and evaluated.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Learning*

Keywords

Classification, k-NN Algorithm, Mode Directed Inverse Entailment

1. INTRODUCTION

The amount of data collected and stored in databases is growing at a very fast pace in almost all areas of human activity. A paramount example is the explosion of biotech data [1, 13], where the volume of data has been doubling every three to six months as a result of automation in biochemistry. Traditional Data Mining approaches search for patterns in a single table (or relation), where each row (or tuple) on the table characterises one object of interest. In more complex applications data may span over several relations, or tables. *Multi-Relational Data Mining* systems can analyse data from multiple relations, with no need to transform the data into a single table first. Several data mining approaches have been proposed, such as tree-mining, graph-mining, or cross-relational mining [12]. One powerful and well-developed abstraction for multi-relational data mining techniques is Inductive Logic Programming (ILP).

To learn patterns in multi-relational data, most ILP approaches rely on Logic Programming languages based on Horn clauses, an important and very useful subset of first-order logic (FOL). Horn clauses can be seen as positive rules. Tables can thus be abstracted as relations, and rules as a form of intensional databases. The expressiveness of FOL gives flexibility and understandability to the induced models. However, ILP is computationally expensive and most ILP systems execute in main memory, thus making scalability a major concern. On complex applications, ILP systems can take several hours, if not days, to return a model. The problem is compounded because, as ILP systems generate queries *dynamically*, it is difficult to segment the database, making space scalability a further concern.

In this work, we propose a two-step multi-relational classification algorithm that integrates a k-Nearest Neighbour algorithm (k-NN) with ILP. Our algorithm, k-RNN, follows the approach proposed in [9] and reduces database scans by performing all accesses in a first step where all queries (or literals) directly related to each example (object) are generated and stored in a compact tree data-structure (e.g., a prefix-tree or *trie*). Database access is only performed when building the trees. We show that such trees are indeed a compact and informative description of the examples by defining a distance measure between these trees, using a k-NN like algorithm to build a classifier, and evaluating its performance. Experimental results on a number of structural activity relationship (SAR) datasets show that k-RNN has both excellent time performance and accuracy.

The remainder of the paper is organised as follows. In Section 2 we present the key concepts that found our work. Section 3 presents the k-RNN algorithm in detail. The empirical evaluation of our proposal is done in Section 4. Section 5 compares our work with related work. Last we conclude and discuss future work.

2. BACKGROUND

The underlying idea of the k-NN algorithm is to classify a new object based on attributes and training samples. The classifiers do not use any model to fit and are only based on memory. Training in k-NN consists of storing all instances into a *training memory*. Actual computation occurs only when a new instance needs to be classified. At this point, the k-NN algorithm scans the instances in the training memory and determines the k instances closest to the instance being

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

classified. Classification is performed using majority vote, where each vote corresponds to the class of each of the k closest objects. Ties can be broken at random.

Traditionally, in a propositional setting each instance is characterised by a fixed set of features. Most often, all features are used to determine the *similarity* between two instances. A standard analogy is to see features as dimensions on an Euclidean space and to estimate similarity between cases (points) using the *Euclidean distance* between the two cases. One drawback of such a measure is that nearest neighbour classification is often particularly vulnerable to irrelevant features: such features may erroneously increase the distance between two similar objects [17]. Also critical for k-NN to work in practise with large data sets is to have good indexing on the training set instances.

Inductive Logic Programming is a major field in Machine Learning with important applications in Data Mining. The fundamental goal of a predictive ILP system is to construct models (usually called hypotheses) given background knowledge B and observations (usually called examples in the ILP literature) E.

Mode-Directed Inverse Entailment (MDIE) [19] is a popular approach in ILP. The key idea in MDIE is to find all literals that could be used in rules that explain the example. This is achieved by selecting a seed example and then constructing the *saturated clause* from the set of all literals that could be used to prove (directly or indirectly) the example. Most ILP systems use the saturated clause in order to bound (anchor) the search space lattice. Therefore, most applications try to have relatively small saturated clauses, as otherwise the search space is as big as if one just enumerates clauses.

The saturated clause can be seen as a directed graph, where literal l_i points to l_j if the clause $l_i \leftarrow l_j$ could be used in a proof of the example. The saturated clause has important theoretical properties. Namely, the clause $\perp(e^{sat}, B, \mathcal{L})$ is the most specific clause within language $\mathcal{L}$ that entails e^{sat}, given B (see [19] for details).

3. THE K-RNN ALGORITHM

In order to apply k-NN in a multi-relational setting, it is fundamental to be able to estimate the distance from an example to others. In MDIE-based learning, all relevant data about an example is in its saturated clause. Therefore, in order to apply k-NN in this setting, we need to be able to measure distance between saturated clauses.

In general, saturated clauses form large directed graphs with up to thousands of nodes. Meaningful distance computation between such graphs is difficult. We therefore follow a different approach. We start from the observation that saturated clauses are used to *generate clauses*. This leads to an intuition for the distance between a new example e_n and an existing example e_o: e_n is more and more different from e_o the more and more different clauses we can generate from the saturated clauses for e_n and e_o. If $T(e)$ is the set of clauses for the example e, a distance measure can be:

$$D(e_n, e_o) = 1 - \frac{\parallel T(e_n) \cap T(e_o) \parallel}{\parallel T(e_n) \parallel}$$

that is, the distance between a new and an old example is a number between 0 and 1. Distance would be 0 if the two examples' bottom-clauses generate the same set of clauses, and would be 1 if they would never generate the same clause.

We can now put our multi-relational classification algorithm that integrates k-NN with predictive ILP together. The algorithm proceeds in two steps:

1. Compile each instance in the training data (positive and negative examples) into a tree [9, 4].
2. For each instance in the test data
   (a) Compile a tree for the instance;
   (b) Calculate the distance between the new tree and all the trees generated from the training data;
   (c) Determine the k nearest neighbours based on the distance measure;
   (d) Gather the category of the nearest neighbours;
   (e) Use simple majority of the category of the nearest neighbours as the prediction value of the query instance.

The algorithm is a direct implementation of k-NN. Most effort will take place in the first, pre-compilation, step (see [9] for more details). For each new example we will then generate a tree, and proceed by computing our distance measure.

Note that unlike the MDIE algorithm referred in the previous section, the k-RNN algorithm generates (compiles) a tree for each positive and negative example. The generation of the trees for each example is performed only once, even in multiple runs such the ones performed while performing cross-validation. This is achieved because the tree representation of each example is kept on disk to be reused whenever needed.

In the implementation of k-RNN, we need a data structure to represent the trees. To do so efficiently, we use tries [10]. To take advantage of the prefix representation of tries, we use an order relation to normalize the literals in a clause before placing it in a trie. Assuming that literal lookup takes constant time when browsing a trie, then trie intersection takes time linear on the size of the smallest trie. In this case, running time is given by the time C required to construct a new trie, representing the unlabeled example, plus the time M required to intersect the new trie with the tries representing the training examples. The time C depends on the maximum clause length, and time M depends on the number of examples and maximum clause length.

4. EXPERIMENTS AND RESULTS

In this section we report the results of an extensive performance study that we conducted to evaluate k-RNN. We performed the study on a number of 2D structure activity datasets. We chose these datasets because they are a practical, relevant application and because they have been extensively studied, allowing for comparison with previous work. Our study addresses several questions. Is k-RNN competitive with state-of-the-art ILP learners? Is k-RNN time and space effective? Is k-RNN sensitive to parameter variation?

4.1 Experimental Settings

Data for the experiments are from well-known biochemical problems in the ILP literature. The first dataset, Carcinogenesis, consists of a set of chemical carcinogens [24]. The second dataset, Mutagenesis, is a dataset of mutagenic nitroaromatics [25]. The dataset includes 2-D structural information on compounds, that is, atom and bond information, plus a number of attributes of molecules. The third dataset, NCTRER, is a more recent dataset extracted from EPA's DSSTox NCTRER database [16]. The Carcinogenesis and Mutagenesis datasets were downloaded from the Machine Learning repositories at the Universities of Oxford[1] and York[2]. The NCTRER dataset was kindly obtained from the Leuven Machine Learning research group [16].

Table 4.1 characterises the datasets in terms of number of positive and negative examples as well as background knowledge size (number of relations used) and total number of tuples (in thousands). We refer the reader to the references cited for details of the datasets and the background knowledge available for each problem.

Table 1: Datasets characterisation where $|E^+|$ is the number of positive examples, $|E^-|$ is the number of negative examples, $\mid B \mid$ is the number of relations, and $\mid Tuples \mid$ is the approximate total number of tuples in the background knowledge (in thousands).

dataset	$\mid E^+ \mid$	$\mid E^- \mid$	$\mid B \mid$	$\mid Tuples \mid$
Carcinogenesis	202	174	44	24 k
Mutagenesis	136	69	21	15 k
NCTRER	131	101	5	15 k

Apart from the obvious differences in the actual chemicals involved, the problems have a number of distinguishing features from a learning point of view. The Mutagenesis problem is known to have good short clauses (with four literals or less) that entail the observed data. The Carcinogenesis problem is known not to have any good short clauses that entail the data (this is purely based on chemical structure: see [6]). Much less is known about the NCTRER dataset.

The three problems pose very different challenges. The Mutagenesis has been thoroughly studied but results vary widely. Standard ILP systems achieve accuracies up to 80%; accuracies over 90% and up to 95% have been achieved either using the DISTILL classifier and using a combination of proposionalisation and SVM based classifiers [18]. Graph based learners such as SUBDUE-CL have also been applied to this task, with results around 80%. To the best of our knowledge, Carcinogenesis is much harder: ILP systems tend to fare around 65%, and only recently a result of 75% was claimed [26]. Recent results for NCTRER indicate that one can achieve up to 80% accuracy using a dynamic combination of ILP and propositional classifiers [15, 16].

For each dataset, we ran the proposed k-RNN algorithm and varied the maximum depth of the clauses that may be placed in the tree (trie) from 2 (one literal in the body) up to 5. We also varied parameter k from 1 up to 20. For all parameter combination we performed a 10-fold cross validation to obtain the averaged accuracy and training time.

The experiments were made on an AMD Athlon(tm) MP 2000+ dual-processor PC with 2 GB of memory, running the Linux Fedora (kernel 2.6.12) operating system.

4.2 Results

We first address the question regarding whether the algorithm is time and space effective. The first step in the execution of the algorithm is to compute each example's trie. This step takes time exponential on the size of the saturated clause, so it should be critical as performance is concerned.

Table 2 presents the pre-processing time, in seconds, taken for each dataset and the average number of clauses compiled and placed in the trie assuming clause length $l = 4$. Notice that this step can be seen as a *compilation* step [9]. It needs to be performed only once as long as saturation-related settings are not changed or the clause length used is not increased.

Table 2: Pre-processing time, average number of clauses compiled *per* example (in thousands) and number of entries in the trie example for a clause length (l) of 4.

dataset	Time (sec)	Clauses	Trie entries
Carcinogenesis	2,840	51 k	2,234
Mutagenesis	6,054	61 k	1,632
NCTRER	368	7 k	520

Table 2 shows that trie generation takes 2,840 seconds in Carcinogenesis and about two times as much in Mutagenesis. Higher times in Mutagenesis are explained by the use of more computationally complex background knowledge, so the extra time is spent on saturated clause generation, not on trie generation. Both systems generate about the same order of candidate clauses, most of them will be discarded. Generating, normalising and discarding the clauses is extremely fast.

Finally, Table 3 presents the average accuracy and average classification time for our approach on the three datasets. We used $k = 4$ as this seems a popular compromise in k-NN, and we used $l = 4$ based on our previous work on using tries to improve the performance of standard ILP approaches [9].

Table 3: Average classification time (per example) and accuracy for k=4 and l=4.

dataset	Time (sec)	Accuracy (%)
Carcinogenesis	3.2	76.84
Mutagenesis	1.6	89.31
NCTRER	0.7	66.95

As far as classification time is concerned k-RNN has a good performance. However, we are aware that there are a lot of space for improvement. At this time, to classify an example the trie of the example is compared with the tries of the training examples, which is not very efficient and does not scale for large number of examples. A way to solve this problem would be to index tries so that the number of comparisons is reduced. This is a open problem.

The results for Carcinogenesis are excellent, to our knowledge they seem to be an improvement over previously reported results in the literature. Results on Mutagenesis are also quite good and comparable to the best results achieved in this dataset: the extensive discussion reported in Lodi and Muggleton [18], indicates k-RNN to be in the top tier in terms of performance, and significantly above state-of-the-art systems such as $kFOIL$ and $nFOIL$ that only achieve around 80%. Last, we can compare our results on NCTRER to the results presented in Landwehr et al [16]. k-RNN outperforms ALEPH [23], which achieved about 50% on this dataset. Performance is comparable to the accuracy obtained by combining of features obtained C-ARMR [21], a relational query miner, with a SVM classifier: k-RNN achieves 69% versus 65% for the ARMR based algorithm.

[1] http://www.comlab.ox.ac.uk/oucl/groups/machlearn/

[2] http://www.cs.york.ac.uk/mlg/index.html

Last, both *kFOIL* and *nFOIL* perform at about 78% accuracy, thus outperforming k-RNN in this dataset.

4.3 Parameter Analysis

A first step in order to understand these results is to analyse how k-RNN performance depends on the two parameters: clause length l and neighbour size k. Tables 4, 5 and 6 present the average accuracy for each dataset for different k and l.

Table 4: Carcinogenesis: Average accuracy (for different k values and clause length l).

l	k=1	k=2	k=3	k=4	k=5	k=10	k=15	k=20
2	53.17	57.63	57.00	58.48	57.00	59.39	62.54	64.66
3	56.74	56.15	62.00	62.37	58.44	62.34	62.32	62.28
4	72.48	72.18	75.92	76.84	74.19	77.76	80.13	79.66
5	58.25	57.65	60.81	61.44	60.24	62.90	62.91	62.06

Table 5: Mutagenesis: Average accuracy (for different k values and clause length l).

l	k=1	k=2	k=3	k=4	k=5	k=10	k=15	k=20
2	80.36	79.31	76.67	74.62	74.56	76.67	77.09	75.45
3	85.11	88.25	85.14	85.64	81.97	79.37	80.03	73.48
4	87.78	87.78	89.31	89.31	87.25	87.36	85.72	80.98
5	87.75	86.64	85.64	86.11	81.95	82.00	78.42	76.37

Table 6: NCTRER: Average accuracy (for different k values and clause length l).

l	k=1	k=2	k=3	k=4	k=5	k=10	k=15	k=20
2	42.69	41.02	42.29	42.29	43.53	68.69	62.61	52.19
3	50.45	53.53	54.00	52.26	55.23	54.39	50.45	51.72
4	67.82	68.69	69.56	66.95	67.82	64.35	62.61	62.61
5	66.87	67.30	68.10	66.83	65.13	65.53	60.34	61.65

The results show that Carcinogenesis is very sensitive to the length of the clauses considered: performance starts from a relatively low result with single literal rules ($l = 2$), does not seem to improve much with two literals, and then improves very significantly when considering rules with three literals in the body ($l = 4$). A similar situation is observed in NCTRER: performance improves with higher values of l. In contrast, the Mutagenesis dataset does not benefit at all from considering complex clauses: the performance is similar for clauses with two, three, or four literals. This seems to suggest that the Mutagenesis dataset is essentially being used as a propositional dataset: the knowledge construction work for the dataset seems to have been very successful in finding useful attributes, and there is little to gain from the structure.

Regarding how k-RNN performance depends on the number of neighbours considered, Carcinogenesis seems to be by far the most sensitive algorithm to variations in k. Performance tends to increase as we increase k from 3 to 20 and it seems to be stabilise for very high k. The other two datasets perform best for relatively small values of k, and performance quickly decreases as k grows.

4.4 Discussion

We study the performance of k-RNN in three different datasets. Our results show k-RNN to be indeed quite competitive with state-of-the art ILP systems, and even able to outperform them on difficult datasets.

Our results also show k-RNN to be effective in terms of time complexity. The main issue in this algorithm is tree construction. Our results show that, for reasonable l (clause length), tree construction is quite fast. Although time grows linearly with the number of examples, trees are constructed independently, suggesting that the algorithm is easily parallelisable. Given the high redundancy in Inductive Logic Program, actual tree sizes are not that large, making tree intersection a fast operation.

Our results also show k-RNN not to be particularly sensitive to k parameter variation. Last, k-RNN provides interesting insight on the multi-relational datasets we analyse. First, it makes it very clear whether learning from a dataset actually benefits from complex relations or not. This is not often as obvious with other approaches. Second, distance provides a different insight into datasets, which can be used to study properties of interest. Both Carcinogenesis and NCTRER show a number of smaller clusters, so it may be of interest to learn rules that distinguish such clusters.

5. RELATED WORK

The saturated clause can be seen as an extensive description of an example. Given that we have such a description, it makes sense to compare different saturated clauses and calculate distances. The problem of finding good distance measures is an old problem in multi-relational data-mining. The DISTILL work from Sebag [22] proposed Distance induction precisely towards k-NN. This technique maps clauses into a Euclidean space. A set of initial d (a system parameter) hypotheses are used to induce the mapping between the hypothesis language and the Euclidean space. The technique was implemented in the STILL system and achieved an accuracy of 96.7% in the Mutagenesis data set. Alternatively, RIBL [8] defines distance by considering sets of facts related to the saturated clause, and then computing distance between them. A generalisation of RIBL could compute distances between recursive terms, including lists [14]. Distances are also fundamental when designing kernels for structured data. Gaertner *et al.* discuss a number of such kernels [11]: to a first approximation, these can be seen as kernels on trees that operate by descending the tree recursively.

To an extent, our approach can also be seen as similar to proposionalisation, as used in Linus [7] or in RSD [28], amongst many others. In such systems, one generates clauses and for each clause finds which examples are covered (so one works from clauses to examples, instead of examples to clauses). Recent work in systems such as SAYU [3], nFOIL [15], and kFOIL [16] address some of the problems in propositionalition by doing greedy search in the space of clauses. SAYU and nFOIL use a Bayesian classifier, but kFOIL uses a kernel computed by calculating which examples match, which we believe is close to our idea of using the intersection to compute distances.

Tries are an important data structure in data mining, where they are often used to efficiently implement algorithms such as apriori [27]. Nijssen and Joost first applied the idea in a multi-relational setting [20] by applying tries to represent the itemsets manipulated by the WARMR algorithm [5].

6. FINAL REMARKS

We proposed a new method for multi-relational data-mining based on first calculating trees for individual instances, and then computing distances between such trees. Our approach uses tree intersection to estimate distances. Results show very good performance on a number of datasets, clearly comparable with state-of-the-art data-mining systems.

An important advantage of our work is that it relies on the same bias language as common ILP systems such as Aleph and Progol. Moreover, our approach can be easily parallelisable, as most execution time is spent in an initial compilation stage, which runs independently for every example.

Our results suggest a number of directions for further work. k-NN tends not to perform well on imbalanced datasets, suggesting that other classification algorithms should be considered. NCTRER suggests that one may need to consider much longer clauses in some datasets. Last, for large datasets one should also implement indexing over the examples' trees (a tree of trees). This said, we believe based on the results observed that k-RNN provides an exciting and robust novel approach to data-mining, that can provide good classification quality and some interesting insights on multi-relational datasets.

7. ACKNOWLEDGMENTS

This work has been partially supported by projects Myddas (POSC/EIA/59154/2004), JEDI (PTDC/EIA/66924/2006), STAMPA (PTDC/EIA/67738/2006) and ILP-Web-Service (PTDC/EIA/70841/2006) and by Fundação para a Ciência e Tecnologia. Nuno A. Fonseca is funded by FCT grant SFRH/BPD/26737/2006.

8. REFERENCES

[1] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. Genbank. *Nucleic Acids Research*, 33:235–242, 2005.

[2] H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele. Improving the efficiency of Inductive Logic Programming through the use of query packs. *JMLR*, 16:135–166, 2002.

[3] J. Davis, E. S. Burnside, I. de Castro Dutra, D. Page, and V. S. Costa. An integrated approach to learning bayesian networks of rules. ECML, volume 3720 of *LNCS*, 84–95. Springer-Verlag, 2005.

[4] R. Camacho, Nuno A. Fonseca, V. Santos Costa, and R. Rocha. ILP :- Just Trie It. *Proceedings of the 2007 International Conference on ILP*, volume 4455 of *LNAI*, 184–198. Springer-Verlag, 2007.

[5] L. Dehaspe and H. Toivonen. *Relational Data Mining*, chapter Discovery of relational association rules, 189–208. Springer-Verlag, 2000.

[6] L. Dehaspe, H. Toivonen, and R. King. Finding frequent substructures in chemical compounds. *International Conference on Knowledge Discovery and Data Mining*, 30–36. AAAI Press, 1998.

[7] S. Džeroski and N. Lavrač. Learning relations from noisy examples: An empirical comparison of LINUS and FOIL. *International Workshop on Machine Learning*, 399–402. Morgan Kaufmann, 1991.

[8] W. Emde and D. Wettschereck. Relational instance based learning. *ICML*, 122–130. Morgan Kaufmann, 1996.

[9] N. A. Fonseca, R. Rocha, R. Camacho, and V. Santos Costa. ILP: Compute Once, Reuse Often. *Workshop on Multi-Relational Data Mining*, 34–45, 2007.

[10] E. Fredkin. Trie Memory. *Communications of the ACM*, 3:490–499, 1962.

[11] T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.

[12] J. Han and M. Kimber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2007.

[13] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N.Shindyalov, and P.E.Bourne. The protein data bank. *Nucleic Acids Research*, 235–242, 2000.

[14] M. Kirsten, S. Wrobel, and T. Horvath. Distance based approaches to relational learning and clustering. 213–230, 2000.

[15] N. Landwehr, K. Kersting, and L. D. Raedt. nfoil: Integrating naïve bayes and foil. *National Conference on Artificial Intelligence*, 795–800, 2005.

[16] N. Landwehr, A. Passerini, L. D. Raedt, and P. Frasconi. kfoil: Learning simple relational kernels. *National Conference on Artificial Intelligence*, 2006.

[17] P. Langley and W. Iba. Average-case analysis of a nearest neighbor algorithm. *IJCAI*, 889–894, 1993.

[18] H. Lodhi and S. H. Muggleton. Is Mutagenesis Still Challenging. *ILP - Late-Breaking Papers*, 35. MIT Press, 2005.

[19] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[20] S. Nijssen and J. N. Kok. Faster association rules for multiple relations. *IJCAI*, 891–896, 2001.

[21] L. D. Raedt and J. Ramon. Condensed representations for inductive logic programming. *Principles of Knowledge Representation and Reasoning*, 438–446. AAAI Press, 2004.

[22] M. Sebag. Distance induction in first order logic. *International Workshop on ILP*, 264–272, 1997.

[23] A. Srinivasan. The Aleph Manual, 2003.

[24] A. Srinivasan, R. D. King, S. Muggleton, and M. J. E. Sternberg. Carcinogenesis predictions using ILP. *International Workshop on ILP*, volume 1297, 273–287. Springer-Verlag, 1997.

[25] A. Srinivasan, S. Muggleton, R. King, and M. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. *International Workshop on ILP*, volume 237 of *GMD-Studien*, 217–232, 1994.

[26] T. Karunaratne and H. Bostroem. Using Background Knowledge for Graph Based Learning: a Case Study in Chemoinformatics. *ILP*, 116–118, 2006.

[27] Y. K. Woon, W. K. Ng, and E. P. Lim. Support-ordered trie for fast frequent itemset discovery. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):875–879, 2004.

[28] F. Zelezný and N. Lavrac. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62(1-2):33–63, 2006.