Classification Inductive Rule Learning with Negated Features

Stephanie Chua, Frans Coenen, and Grant Malcolm

Department of Computer Science,
University of Liverpool, Ashton Building,
Ashton Street, L69 3BX Liverpool, UK.
{s.chua,coenen,grant}@liverpool.ac.uk

Abstract. This paper reports on an investigation to compare a number of strategies to include negated features within the process of Inductive Rule Learning (IRL). The emphasis is on generating the negation of features while rules are being "learnt"; rather than including (or deriving) the negation of all features as part of the input. Eight different strategies are considered based on the manipulation of three feature sub-spaces. Comparisons are also made with Associative Rule Learning (ARL) in the context of multi-class text classification. The results indicate that the option to include negated features within the IRL process produces more effective classifiers.

Key words: Rule Learning, Negation, Multi-class Text Classification

1 Introduction

The generation of rule-based classifiers is a popular data mining method which is applicable to many data formats. Its popularity is largely due to its *transparency*, the ease with which classifications can be explained, and its relative simplicity (compared to other types of classifiers such as support vector machines or neural networks).

Rules in rule-based classifiers are typically represented in the form $X \Rightarrow k$, where X is some subset of the global set of features (A) present in the input data, and k is a class label. For example, we might have a rule $a \wedge b \Rightarrow k$ (where a and b are two features taken from A). The rule is interpreted as, "if a and b exist in a record, then classify the record as belonging to class k". Examples of rule-based classifiers include Associative Rule Learning (ARL) systems such as Classification based on Multiple class-Association Rules (CMAR) [12], Classification based on Predictive Association Rules (CPAR) [17], or Total From Partial Classification (TFPC) [5] and Inductive Rule Learning (IRL) systems such as First Order Inductive Learner (FOIL) [13] and Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [6].

Many rule-based classifiers such as CPAR, CMAR, TFPC and FOIL do not provide the option to include rules with negated features of the form: $a \wedge b \wedge \neg c \Rightarrow k$. Note that such a rule would be interpreted as, "if a and b exist in a record and

c does not, then classify the record as belonging to class k". There are only a few examples of rule-based classifier generation systems which generate rules with negation. One such system is the Olex system [14]. However, this system uses a very restrictive rule template (one positive feature and zero or more negative features) for rule generation and that significantly restricts the nature of the rules that can be generated.

Of course, negation can be included explicitly in the input data by including a negated feature for every feature that exists (for example, a and $\neg a$). It is also possible to generate negated versions for all features automatically as part of the classifier generation process. For some applications, this is a realistic option. However, for many applications, this is not realistic because it doubles the number of features to be considered. One example is in the field of text classification where documents are encoded using the *bag-of-words* representation. Using this representation, the feature set comprises several thousand words and thus, doubling the number of features would introduce an additional computational overhead (the computational complexity of rule-based classifier generators increases exponentially with the number of features).

The approach advocated in this paper is to include negated features within the rule generation process without explicitly encoding such negations in the input or generating all possible negations. In this case, there are two issues to be addressed: (i) the process of identifying features whose negation and inclusion in a rule under consideration during the rule generation process can benefit the final result (the accuracy of the classifier), and (ii) the rule refinement strategies to dictate whether a positive feature or a negative feature should be included in a rule. In this paper, a number of strategies are proposed and evaluated for addressing both these issues. The focus of this work is on multi-class text classification using the bag-of-words representation. However, the reported investigation is equally applicable to other forms of classification.

The rest of this paper is organized as follows. Section 2 describes some previous reported work concerning rule learning with and without negation. Section 3 discusses the adopted rule-based text classification model, followed by the proposed inductive rule learning algorithm in Section 4. The experimental setup is presented in Section 5. The results and an analysis of the experiments are discussed in Section 6. Section 7 concludes this paper with a brief discussion of future work.

2 Previous Work

Many machine learning methods for text classification have been proposed. Inductive Rule Learning (IRL) methods tend to be based on the covering algorithm. In the covering algorithm, rules are learned one at a time based on the training examples given. The examples "covered" by a rule are then removed and the process is repeated until a stopping condition is met. Examples of IRL systems based on the covering algorithm include: (i) Incremental Reduced Error Pruning (IREP) [7], (ii) Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [6] and (iii) Swap-1 [16, 3]. An example of the use of negation in IRL can be found in the Olex system [14] which uses positive and negative features to generate rules using a template of one positive feature and none or more negative feature(s). However, the use of this template is very restrictive in that no two positive features can co-occur together in a rule.

ARL algorithms are based on the concept of Association Rule Mining (ARM) as first proposed by [1]. ARM operates by generating the set of relationships between features that exist in a given data set. These relations are expressed as probabilistic rules, called Association Rules (ARs). ARs are of the form $X \Rightarrow Y$ (where X and Y are disjoint subsets of the the global set of features A). ARs are interpreted as "if X exists in a record then it is likely that Y will also exist". The set of classifcation rules that exist in a data set are a subset of the set of ARs. A number of ARL systems were identified in Section 1. Negation has been used in previous work in ARL. For example, Antonie and Zaïane [2] proposed an algorithm that discovers negative ARs (one example being, "if X exists in a record, then it is likely that Y will NOT exist". Baralis and Garza [4] constructed an associative classifier that used rules with negated words. They reported that the use of negated words improved the quality of their classifiers. A criticism of the ARL approach is that a great many rules are generated, typically many more than in the case of IRL systems such as the approach proposed in this paper. However, to evaluate the IRL approach proposed in this paper, comparisons are made with the ARL technique using the TFPC algorithm [5] because we are interested in interpretable rules as a classifier and also because our experiments here focus on multi-class classification instead of binary classification. Therefore, we will not compare with the support vector machine (SVM) [9] method (reported to be one of the best method for text classification).

3 Rule-based Text Classification Model

A general model for a rule-based text classification consists of a number of processes (Figure 1). These processes include preprocessing, text representation and rule learning. To evaluate the generated classifier, the input data (document base) is usually split into a training set and a test set. The first step is to preprocess the data. There are many sub-processes that may be applied at this stage, such as stop word removal, stemming and feature selection. After preprocessing has been completed, the documents in their preprocessed form are translated into some appropriate representation suitable for the application of text classification algorithms. A popular method for text representation is the *bag-of-words* representation. In this representation, a document is represented in an N-dimensional vector space, where N is the number of features. The value of each feature in the vector space can take either a Boolean value to indicate the presence or absence of a word in a document or a numeric value to indicate its frequency in a document. Note that when using the bag-of-words representation, information regarding the order and position of the words is lost.

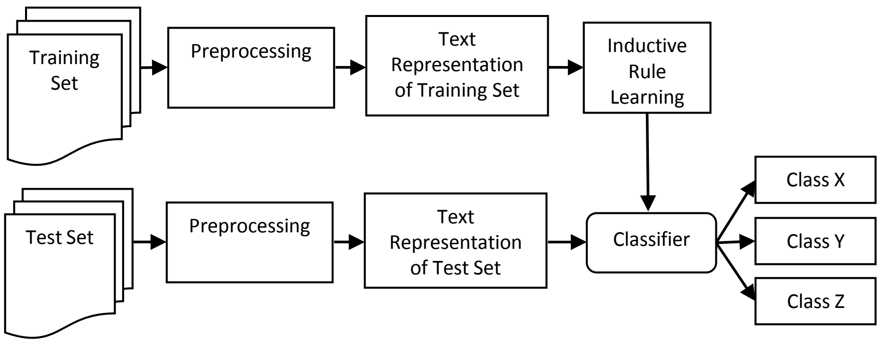

Fig. 1. Rule-based text classification model

Once the appropriate representation has been generated, the IRL process can be applied. During this process, the desired set of classifcation rules are learned, resulting in a classifier which may be applied to “unseen” data.

4 Inductive Rule Learning with Negation

The algorithm for our inductive rule learner is presented in Table 1. Our inductive rule learner is founded on the sequential covering algorithm. For a given class k, rules are learned sequentially one at a time based on training examples using the **LearnOneRule** method (Table 2). The examples “covered” by a rule learnt are then removed and the process is repeated until some stopping condition is met. Stopping conditions are: (i) when there are no more uncovered documents or (ii) when there are no more unused features in the feature set. Each rule generated is the ruleset *so far*. Post-processing is done on the final ruleset using the **PostProcess** method to remove rules with a rule accuracy, defined in our case using the Laplace estimation accuracy, lower than a user pre-defined threshold.

Rules are generated using the *specialization* approach, whereby a rule is made more specific by adding features to its *condition*. In the **LearnOneRule** method, rule learning starts with an empty *condition* and a class c in the *conclusion*. The top most unused feature from the $Feature\_set$ is added to the empty rule and the rule is checked using the **CheckRule** method (Table 3). In the **CheckRule** method, the rule “coverage” is checked, i.e. the number of positive and negative documents that it covers. If a rule covers negative document(s) and it can be further refined, then the **RefineRule** method is called. There are a number of different strategies for refining rules presented in this paper. These strategies for refinement are what make our IRL system different from previously proposed systems. While previous methods used pruning (e.g.[7, 6]) to improve their rule quality, our approach uses refinement strategies that are applied whilst the rule is being generated. Rules with and without negation are

Table 1. Algorithm for inductive rule learner (adapted from [8])

```
Algorithm: Learn a set of IF-THEN rules for classification.

Input:
D, a dataset of class-labelled documents;
Feature_set, the set of features for class c;

Output: A set of IF-THEN rules.

Method:
Rule_set = { }; //initial set of rules learned is empty

for each class c do
    repeat
        Rule = LearnOneRule(Feature_set, D, c);
        remove documents covered by Rule from D;
        Rule_set = Rule_set + Rule; //add new rule to ruleset
    until stopping condition;
endfor
PostProcess(Rule_set);
return Rule_set;
```

Table 2. Algorithm for LearnOneRule method

```
Algorithm: LearnOneRule.

Input:
D, a dataset of class-labelled documents;
Feature_set, the set of features for class c;
c, class label;

Output: Rule

Method:
Create a new empty rule, Rule
Set Rule condition to c
Add top most unused feature from Feature_set
Rule = CheckRule(Rule, D)
return Rule
```

generated based on the usage of search space division, as described in Section 4.1.

The model was implemented with a view to multi-class classification, i.e. the generation of a rule-based classifier designed to classify unseen cases into one of N classes. Other classifiers include binary classifiers, which assign an unseen case as belonging to a class or not and classifiers that can assign more than one class to an unseen case. A sequence of binary classifiers is usually used to achieve multi-class classification in most previous work. In binary classification, the classifier has only rules from one class and these rules are used to classify documents as belonging to the particular class or not. In contrast, in multi-class classification, the classifier consists of rules from all the classes in the dataset. In order to determine which rule is to fire when classifying a document, the rules have to be ordered. The higher order rules will be fired before lower ones. In our experiments (see Section 5), the rules were first ordered according to descending rule length (the number of features in the rule *condition*) so that more specific rules would be fired first; and then, if the rules were of the same length, then the descending rule weight (the number of documents the rule covered in the learning phase) will be used for ordering. This meant that more specific rules with higher coverage would rank higher than less specific and lower coverage rules.

Table 3. Algorithm for CheckRule method

```
Algorithm: CheckRule

Input:
D, a dataset of class-labelled documents;
Rule, the rule to check;

Output:  Rule

Method:
Check rule coverage


If Rule covers negative document(s)
        If Rule can be further refined
                Rule = RefineRule(Rule)
return Rule
```

4.1 Rule Refinement

During the proposed IRL process, the construction of each rule commences with an initial “start” rule comprising a single positive feature in its *condition* and an associated class in the *conclusion*. This single positive feature is selected from the top most unused feature in the feature set, ranked in descending order of the chi-square value of the features. If the rule *so far* covers both positive and negative documents, then the rule has to be refined in order to learn a rule that can separate the positive and negative documents. If the rule covers only positive documents, then it is added to the ruleset and the process continues with the generation of the next rule. During the refinement process, an appropriate feature is selected from the search space to add to the rule. The search space contains features from both the positive and negative documents that are covered by the rule. The search space can thus be divided into three sub-spaces that contain different kinds of feature:

- **Unique postive** (UP) features which are only found in positive documents.
- **Unique negative** (UN) features which are only found in negative documents.
- **Overlap** (Ov) features which are found in both positive and negative documents.

Such a division allows for effective and efficient identification of both positive features and features that can advantageously be negated. It should be noted that the UP, UN and Ov sub-spaces may be empty, as the existence of these features is dependent upon the content of the documents covered by a rule. When refining a rule, a feature from either the UP, UN and Ov feature sub-spaces can be selected to be added to the rule *so far*. If a rule is refined with a UP or Ov feature; then a new rule *so far*, with no negation, is generated. If a rule is refined with a UN feature, then the rule *so far* will include a negated feature. The Ov feature is added as a positive feature despite appearing in both positive and negative documents because negating an Ov feature in a rule will generate a rule that rejects positive documents.

Eight different rule refinement strategies were devised with respect to the three identified sub-spaces. These strategies are presented in Table 4. Note that strategies UP, Ov and BestPosRule generate rules without any negated features and have been included for comparison purposes. Strategies UN and UN-UP-Ov will always generate rules with negation provided the UN sub-space is not empty. Note that the UN strategy will result in rules comprising a positive feature and one or more negative features; thus a generating rules with a template identical to the Olex system [14] described in Section 2. Strategy UP-UN-Ov will generate rules without negation provided UP is not empty. The BestStrategy strategy will choose the best rule from the rules generated using the UP, UN, Ov, UP-UN-Ov and UN-UP-Ov strategies and thus may generate rules with negation. The BestRule strategy may also generate rules with negation, provided that when adding a UN feature, a better rule is generated than when adding a UP or Ov feature.

Table 4. Rule refinement strategies

Strategy	Sub-space used	Description
UP	UP	Add a UP feature to refine a rule
UN	UN	Add a UN feature to refine a rule
Ov	Ov	Add an Ov feature to refine a rule
UP-UN-Ov	Either a UP, UN or Ov feature in every round of refinement	If UP is not empty, add a UP feature to refine a rule; Else If UN is not empty, add a UN feature to refine a rule; Else If Ov is not empty, add an Ov feature to refine a rule.
UN-UP-Ov	Either a UP, UN or Ov feature in every round of refinement	If UN is not empty, add a UN feature to refine a rule; Else If UP is not empty, add a UP feature to refine a rule; Else If Ov is not empty, add an Ov feature to refine a rule.
BestStrategy	All sub-spaces	Choose the best rule from the five rules generated by each UP, UN, Ov, UP-UN-Ov and UN-UP-Ov.
BestPosRule	UP and Ov in every round of refinement	Generate two versions of rule; one refined with a UP feature and the other refined with an Ov feature. Choose the best between the two versions.
BestRule	UP, UN and Ov in every round of refinement	Generate three versions of rule; one refined with a UP feature, one refined with a UN feature and the other refined with an Ov feature. Choose the best between the three versions.

5 Experimental Setup

The data sets used were the 20 Newsgroups data set [10] and the Reuters-21578 data set [11]. The 20 Newsgroups data set consists of 20 classes and 19,997 documents, while the Reuters-21578 data set consists of 135 classes and 21,578 documents. In our preparation of the data sets, we adopted the approach of Wang [15]. In his work, he split the 20 Newsgroups data set into two non-overlapping data sets (*20NG-A* and *20NG-B*), each containing 10 classes. Each class contained 1,000 documents with the exception of one class in *20NG-B* that had 997 documents. In the Reuters-21578 data set, the top ten most populous classes were first selected and multi-labelled and empty documents (documents that did not include any text) were removed, leaving a data set with eight classes and 6,643 documents, hereafter referred to as *Reuters8*.

The rule-based text classification model described in Section 3 was applied to all the data sets. Preprocessing included stop words removal and feature selection. Chi-square was used in a local feature selection environment with a reduction factor of 0.9 (only 10% of the features from each class were used). The proposed rule refinement strategies were applied during the rule learning stage. Rules with a Laplace estimation accuracy lower than the threshold of 50% were removed from the ruleset. The performance of our inductive rule learner, with each of the different proposed rule refinement strategies, was compared with the TFPC associative rule learner of Coenen and Leng [5]. The evaluation metric used is the average accuracy produced using ten fold cross validation.

6 Results and Analysis

The experiments conducted compared our inductive rule learner, and its different rule refinement strategies, with an associative rule learner in a multi-class classification setting. Hereafter, our inductive rule learner is denoted as RL appended with the identifier of the different rule refinement strategies, and the associative rule learner is denoted as ARL. Table 5, 6 and 7 show the average accuracy obtained using ten fold cross validation, the average number of rules generated, the average number of rules with negation generated and the percentage of rules with negation, for the *20NG-A*, *20NG-B* and *Reuters8* data sets respectively. In each case, the highest accuracy is highlighted in bold.

Table 5. Results using the 20NG-A dataset

Method	Avg accuracy	Avg # rules generated	Avg # rules with negation generated	Avg % of rules with negation
RL + UP	78.3	218.0	0.0	0.0
RL + UN	73.3	132.0	46.0	34.8
RL + Ov	77.1	194.2	0.0	0.0
RL + UP-UN-Ov	78.3	218.0	0.0	0.0
RL + UN-UP-Ov	77.8	218.9	76.1	34.8
RL + BestStrategy	77.7	199.9	37.8	18.9
RL + BestPosRule	78.5	218.9	0.0	0.0
RL + BestRule	**79.8**	194.4	48.6	25.0
ARL	76.0	1582.9	0.0	0.0

With respect to classification accuracy, inspection of the results indicated that the best performing method in all cases was the RL + BestRule strategy. RL + UN performed worst in the *20NG-A* dataset, RL + Ov performed worst in the *20NG-B* dataset, while ARL and RL + UN performed equally worst in the *Reuters8* dataset.

With respect to the overall number of rules generated in each case, ARL produced many more rules than any of the RL strategies. This is because ARL approaches, such as TFPC produce many frequent *item sets* (patterns) from which classification rules are generated. The number of rules generated by ARL

Table 6. Results using the 20NG-B dataset

Method	Avg accuracy	Avg # rules generated	Avg # rules with negation generated	Avg % of rules with negation
RL + UP	79.3	216.8	0.0	0.0
RL + UN	79.9	110.3	47.2	42.8
RL + Ov	78.7	196.4	0.0	0.0
RL + UP-UN-Ov	79.3	216.8	0.0	0.0
RL + UN-UP-Ov	79.8	209.6	81.0	38.6
RL + BestStrategy	79.5	161.1	45.2	28.1
RL + BestPosRule	79.0	217.5	0.0	0.0
RL + BestRule	**81.2**	169.9	53.0	31.2
ARL	79.2	1546.1	0.0	0.0

Table 7. Results using the Reuters8 dataset

Method	Avg accuracy	Avg # rules generated	Avg # rules with negation generated	Avg % of rules with negation
RL + UP	85.0	133.0	0.0	0.0
RL + UN	75.2	25.6	25.6	100.0
RL + Ov	78.0	105.9	0.0	0.0
RL + UP-UN-Ov	85.0	132.9	0.0	0.0
RL + UN-UP-Ov	89.1	121.6	108.6	89.3
RL + BestStrategy	89.0	99.5	58.9	59.2
RL + BestPosRule	85.1	129.1	0.0	0.0
RL + BestRule	**90.3**	97.3	66.9	68.8
ARL	75.2	3235.6	0.0	0.0

systems is a criticism frequently directed at this approach. With respect to the eight strategies, there is no significant variation with respect to the number of rules generated although the *Reuters8* data set seems to require fewer rules than the *20NG-A* and *20NG-B* data sets. Recall that RL + UP, RL + Ov and RL + BestPosRule cannot generate rules with negative features. In addition, inspection of the results shows that RL + UP-UN-OV also does not generate rules with negative features, suggesting that the UP sub-space was never empty with respect to all three data sets and thus, produces the same classification results as the RL + UP strategy. The relatively poorer performance of the RL + UN method is attributed to the structure of the rules generated, in that rules with negation generated by this strategy take the structure of “one positive feature and one or more negative feature” (i.e. identical to the Olex system [14]). The disadvantage of this rule structure is that having only one positive feature in its rule condition (despite having one or more negative features) makes the rule overly general in that the single positive feature is prone to misclassifying documents which the negative features cannot overturn. Also, it is worth noting that RL + UN generates fewer rules than any of the other strategies. This is again attributed to the rule structure in that a single positive feature can generally cover a large portion of documents, and thus, less rules are being learnt. In direct comparison of RL + BestPosRule and RL + BestRule, where the latter is an extension of the former by allowing the inclusion of UN feature in refinement, RL + BestRule outperformed RL + BestPosRule in all three datasets. This strongly suggests that the use of negated features can produce more effective classifiers.

If we look at the number of negative rules generated in the case of strategies that permit negative features, a significant proportion are rules that include negated features. The proportion is higher with respect to the *Reuters8* data set than for the *20NG-A* and *20NG-B* data sets. Overall, the results indicate that the strategies that include negated features in classification rules generated using IRL give a higher accuracy than strategies that do not include negated features.

7 Conclusion and Future Work

In this paper, we have presented an evaluation of a number of strategies to include negated features in IRL systems. These different rule refinement strategies enabled the generation of rules with and without negation. The approach was applied to text classification using the bag-of-words representation. The reported comparison between the different rule refinement strategies demonstrated that RL + BestRule produced the best results in all cases. In general, rule refinement strategies that involved negated features performed well indicating the advantage that can be gained in the context of text classifcation. In addition, a comparison was also made with an Associative Rule Learner (ARL); the proposed variations of the IRL method outperformed ARL in nearly all cases. Moreover, all of the IRL strategies used were able to produce more compact (smaller) classifiers with less rules. In conclusion, the results reported in this paper demonstrate that rules with negation are more effective for the multi-class classification task than rules that rely on positive features only.

Our future work will further extend the current work by using phrases instead of just single keywords in the text classification task. Phrases contain semantic information which, as noted above, are lost in the bag-of-words representation. For example, “Bank of England” and “river bank” are two phrases with entirely different semantic content. We are motivated by the potential benefit of using this semantic information in phrases to provide an added advantage to the text classification task. More specifically, we will also look into the use of negated phrases within the context of the work described in this paper.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 207-216 (1993)

2. Antonie, M-L., Zaïane, O. R.: An associative classifier based on positive and negative rules. In: Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 64-69 (2004)
3. Apté, C., Damerau, F. J., Weiss, S. M.: Automated learning of decision rules for text categorization. In: ACM Transactions on Information Systems 12, 233-251 (1994)
4. Baralis, E., Garza, P.: Associative text categorization exploiting negated words. In: Proceedings of the ACM Symposium on Applied Computing, pp.530-535 (2006)
5. Coenen, F., Leng, P.: The Effect of Threshold Values on Association Rule Based Classification Accuracy. In: Journal of Data and Knowledge Engineering. Vol. 60, Num. 2 (February 2007), pp345-360 (2007).
6. Cohen, W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning (ICML), pp. 115-123, Morgan Kaufmann (1995)
7. Fürnkranz, J., Widmer, G.: Incremental reduced error pruning. In: Proceedings of the 11th International Conference on Machine Learning (ICML), Morgan Kaufmann (1994)
8. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2006)
9. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning (ECML), pp. 137-142 (1998)
10. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th International Conference on Machine Learning, pp. 331-339 (1995)
11. Lewis, D. D.: Reuters-21578 text categorization test collection, Distribution 1.0, README file (v 1.3). Available at `http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt` (2004)
12. Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient Classification based on Multiple class-Association Rules. In: Proceedings of the IEEE International Conference on Data Mining, pp. 369-376 (2001)
13. Quinlan, J. R., Cameron-Jones, R. M.: FOIL: A midterm report. In: Proceedings of the European Conference on Machine Learning (ECML), pp. 3-20, Springer-Verlag (1993)
14. Rullo, P., Cumbo, C., Policicchio, V. L.: Learning rules with negation for text categorization. In: Proceedings of the 22nd ACM Symposium on Applied Computing, pp. 409-416. ACM (2007)
15. Wang, Y. J.: Language-independent pre-processing of large documentbases for text classifcation. PhD thesis (2007)
16. Weiss, S. M., Indurkhya, N.: Optimized rule induction. In: IEEE Expert: Intelligent Systems and Their Applications 8, 61-69 (1993)
17. Yin, X., Han, J.: CPAR: Classification based on Predictive Association Rules. In: Proceedings of the SIAM International Conference on Data Mining, pp. 331-335 (2003)