Rule Learning with Negation for Text Classification

Stephanie Chua and Frans Coenen

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

Abstract. Classification rule generators that have the potential to include negated features in their antecedents are generally acknowledged to generate rules that have greater discriminating power than rules without negation. This can be achieved by including the negation of all features as part of the input. However, this is only viable if the number of features is relatively small. There are many applications where this is not the case, for example text classification. Given a large number of features, alternative strategies for including negated features in rules are desirable. This paper explores a number of strategies whereby this can be achieved in the context of inductive rule learning. Eight different strategies are proposed and evaluated by comparison with JRip, NaiveBayes and SMO. The reported results demonstrate that rules with negation produced a better classifier and that our rule learning mechanism outperform JRip and NaiveBayes and is competitive with SMO.

1 Introduction

Text classification is concerned with the allocation of documents to pre-defined classes according to patterns extracted from a set of labelled training documents. Originally text classification was conducted manually by reading the documents and deciding which "class" they should be assigned to. As technology advanced, expert systems were built, using manually constructed rules to classify documents. One such system was the CONSTRUE system [11] built by the Carnegie Group for the Reuters news agency. However, due to the large increase in digital documents, this method of manual rule construction by experts has become increasingly expensive and consequently, increasingly infeasible. Machine learning methods have thus become a popular approach to text classification.

Many machine learning methods are described in the literature for text classification. Among them are decision trees [12, 14], nearest neighbour methods [22], Support Vector Machines (SVM) [13], probabilistic Bayesian models [17, 22], and neural networks [6]. The technique of interest with respect to this paper is inductive rule learning (IRL) [2, 5]. What all of these methods have in common is that they are used to build a classifier, which can then be used to classify new and unseen records. The generated classifier can be expressed in a number

of ways; IRL results in what is known as a *rule-based* classifier. The advantage of rule-based classifiers over other representations is that they are easily interpretable by humans. In the context of rule-based classifiers, a rule is represented in the following form:

$$condition \Rightarrow conclusion$$

where the *condition* consists of a conjunction of features that occur together, and the *conclusion* is the resulting class label associated with the *condition*. For example, if a, b and c are features that appear in documents within a dataset, and x is a class label, the rule $a \wedge b \wedge c \Rightarrow x$ is interpreted as "if a, b, and c occur together in a given document, then classify the document as belonging to the class x".

Typically, a classifier built using IRL comprises rules that contain only positive features. Rules with negation come in the form of $a \wedge b \wedge \neg c \Rightarrow x$ and would be interpreted as, "if a and b occur together in a given document and c does not, then classify the document as belonging to class x". Negation can be included (and in some cases is included) by generating the negation of every feature in the feature space. However, this will substantially increase the size of the input feature space (in the case of binary-valued feature sets, this will double the number of features). Increasing the size of the feature space makes very little difference given a small numbers of features. However, in the event of having a large number of features, this is not a realistic proposition. This is especially the case if we recall that the number of potential antecedents is $2^n - 1$ (where n is the number of features) and that the complexity of IRL systems typically increases exponentially with the size of n. One category of application that involves a large number of features, and the focus of the work described in this paper, is text classification. Documents for text classification are typically represented using feature vectors, where each vector is some subset of a bag-of-words (or bag-of-phrases). The bag-of-words representation typically comprises many keywords; therefore, including the negation of these words would double the number. A mechanism whereby negation can be included in IRL, without generating all potential negated features, is therefore desirable. This is then the motivation for the work described in this paper.

The desire to include negation in IRL requires the resolution of two issues. The first is the process for identifying appropriate negated features. The second is the decision making process when refining a rule, to decide whether to include a negated feature or a positive feature. There are a number of potential solutions to these issues and these are considered and compared in this paper. The rest of this paper is organized as follows. Section 2 describes some related work on rule learning. Section 3 discusses our proposed mechanism for inductive rule learning with negation. Section 4 considers the negated feature identification issue, and Section 5, the issue of whether to refine a rule with a negated feature or a positive feature. The datasets used are described in Section 6. Section 7 describes the evaluation results. Section 8 concludes the paper.

2 Previous Work

Reported IRL algorithms that have been directed at text classification include: Reduced Error Pruning (REP) [4], Incremental Reduced Error Pruning (IREP) [7], Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [5] and Swap-1 [20]. These algorithms do not explicitly advocate any usage of negated features in the learning of classification rules[1]. Instead, these methods employed a two-stage process where rules are first learnt and then pruned to improve the effectiveness of the ruleset. All the above systems make use of the covering algorithm for rule learning. Rules are usually generated using the sequential covering algorithm where rules are "learned" (sequentially) using a set of training examples. The "covered" examples are then removed and the process is repeated until all the examples are covered.

Rullo et al. [18] proposed a system called Olex that used a single stage rule learning process without pruning. This system is of interest in the context of the work described here because they investigated using both positive and negative features for rule learning. They proposed the paradigm of "one positive term, more negative terms", where each rule generated is a conjunction of a single positive feature with none or more negative features. While they reported positive results in the use of negation, they highlighted that their approach was not able to express co-occurrences, based on feature dependencies, as a result of allowing exactly one positive feature in the rule antecedent [18]. Thus, Olex is unable to generate rules of the form $a \wedge b \wedge \neg c \Rightarrow x$.

A number of alternative methods for incorporating negated features into text classification have been reported in the literature, though not all of these methods are intended for rule-based learning. Antonie and Zaïane [1] and Wu et al. [21] used both positive and negative association rules in their work on classification association rule mining. Galvotti et al. [8] used a novel variant of k-NN with negative evidence. Zheng and Srihari [23] combined positive and negative features in their feature selection method for text classification and used a Naive Bayes classifier. Baralis and Garza [3] used negated words in their associative classifier for text classification. The use of negation in previous work has served to provide empirical evidence of the effectiveness of the incorporation of negated features for text classification, thus providing further motivation for the investigation described in this paper.

3 Inductive Rule Learning with Negation

For the described investigation, a rule learning mechanism founded on the sequential covering algorithm was adopted. A high level view of the algorithm is presented in Table 1. For each class c in the given set of classes C, rules

[1] RIPPER appears to generate rules with negation by including all possible values for each feature during rule learning. Therefore, the case of binary feature c with the value 0 (c=0) would indicate the absence of c, thus, a rule $a = 1 \wedge b = 1 \wedge c = 0 \Rightarrow x$ would be interpreted in a similar manner to that described in Section 1.

are learned sequentially one at a time based on training examples using the **LearnOneRule** method. The examples "covered" by a rule learnt are then removed and the process is repeated until some stopping condition is met. The two possible stopping conditions are: (i) there are no more uncovered documents in the document set, and (ii) there are no more unused features in the feature set. Each rule generated is added to the ruleset "sofar". In the method **PostProcess**, post-processing is applied to the ruleset to remove rules, with a Laplace estimation accuracy of less than a pre-defined threshold, which are deemed to be ineffective.

When a rule covers both positive and negative documents, the rule has to be refined in order to separate the documents with different class labels. In this context, positive documents are documents in the training set that are correctly classified by the current rule while negative documents are documents that are incorrectly classified. Generating rules without negation is straightforward; features that occur together in a positive document are used as conjunctions in a rule to separate the positive and negative documents. On the other hand, generating rules with negation requires the identification of the feature to be negated. This is one of the issues to be addressed in this investigation. Another issue concerns the strategies to refine a rule. When a rule can be refined with both positive and negative features, would it be better to refine the rule with a positive feature, thus generating a rule without negation, or to refine the rule with a negated feature, thus generating a rule with negation? Proposed solutions for addressing these two issues are presented in Sections 4 and 5.

Table 1. Pseudocode for the adopted rule learning mechanism founded on the sequential covering algorithm (adapted from [10])

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*;
    until stopping condition;
endfor
PostProcess(*Rule_set*);
return *Rule_set*;

4 Identifying Features

The goodness or discriminative power of a feature with respect to a class is usually evaluated using some statistical measure. In text classification, measures like chi-square (χ^2) and information gain (IG) are commonly used to statistically evaluate features in the context of feature selection. The χ^2/IG value of a feature is computed with respect to a specific class. The features are ordered in descending order of their χ^2/IG values and a subset of these ordered features are selected for rule learning. There are two strategies for feature selection: local and global. In local feature selection, ordered features local to a specific class are selected for learning a classifier for that specific class. In contrast, global feature selection involves the selection of features from across all classes in a dataset. The maximum or weighted-average value of each feature's class-specific values can then be used for the ordering and selecting of features. In our experiments, despite a rigorous reduction factor of 0.9 (using only 10% of the features), the global feature selection method still entails very high computational power. Therefore, we focus on using the local feature selection method.

In our proposed mechanism, during rule refinement, an appropriate feature is selected from the *search space* to be added to the rule. The search space contains features from both the positive and negative documents that are covered by the rule. This research proposes that this search space be divided into three subspaces that contain different categories of feature:

1. **Unique Postive** (UP). Features that only appear in positive documents.
2. **Unique Negative** (UN). Features that only appear in negative documents.
3. **Overlap** (Ov). Features that are found in both positive and negative documents.

This division allows for the effective and efficient identification of features that can be advantageously negated for the purpose of generating rules with negation. 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 or Ov sub-spaces can be selected to be added to a rule. If a rule is refined with a UP or Ov feature, then a rule with no negation is generated. If a rule is refined with a UN feature, then a rule with negation is generated. When refining a rule with a UP or UN feature, the feature with the highest document frequency, i.e. the feature that appears in the most covered documents, is selected. This ensures that the refined rule will cover the maximum possible number of positive documents at every round of refinement. When refining a rule with an Ov feature, the feature with the highest document frequency difference (i.e. positive document frequency minus negative document frequency) is selected. This is because an Ov feature occurs in both positive and negative documents and the feature that appears in most positive documents and least negative documents will result in a refined rule that covers the maximum possible number of positive documents.

5 Rule Refinement Strategies

Given the three sub-spaces, UP, UN and Ov, a total of eight strategies for rule refinement were devised. The first three strategies were directed at utilising only a single sub-space. The strategies were named after the sub-space used: UP, UN and Ov. Given that a sub-space may be empty, having an empty set using the UP, UN and Ov strategies would mean that refinement may be prematurely halted in the absence of any features to be added. Therefore, two other strategies were devised to address the empty sub-space problem; UP-UN-Ov and UN-UP-Ov. These strategies use a sequence of sub-space combinations and were labelled in the order that the sub-spaces are considered. Thus, the sequence combination of UP-UN-Ov entails the use of UP features first. If the UP sub-space is empty, the UN features will be considered instead, and then the Ov features, if the UN sub-space is also empty. The UN-UP-Ov sequence combination strategy works in a similar manner, only inter-changing the order of UP and UN. In both cases, Ov is used last because using Ov features will always result in the coverage of at least one negative document. In both cases, if the first sub-space is not empty, then only the first sub-space will be used for rule refinement. This would mean that the UP-UN-Ov sequence combination strategy may produce the same results as the UP strategy, and similarly the UN-UP-Ov sequence combination strategy may produce the same results as the UN strategy. The implementation of these five strategies are straightforward and are shown in Tables 2, 3, 4, 5 and 6.

Table 2. RefineWithUP algorithm

Algorithm: RefineWithUP

Input: R, a rule that needs refinement
Output: R', refined rule
Method:

if $UPSubSpace$ is not empty
  $R' =$ **addUPFeature**(R);
else
  $R' = R$;

return R';

For each rule that needs refinement, each of the five afore-mentioned strategies will generate a different final rule. The sixth proposed strategy called Best-Strategy operates on choosing the best rule from the rules generated by the first five strategies. The rule with the highest rule accuracy, using the Laplace estimation, is considered the best rule. The algorithm is shown in Table 7.

A common drawback associated with the first five strategies is that on each rule refinement round, only one type of sub-space is used. This means that in

Table 3. RefineWithUN algorithm

Algorithm: RefineWithUN

Input: R, a rule that needs refinement
Output: R', refined rule
Method:

if $UNSubSpace$ is not empty
  $R' =$ **addUNFeature(**R**)**;
else
  $R' = R$;

return R';

Table 4. RefineWithOv algorithm

Algorithm: RefineWithOv

Input: R, a rule that needs refinement
Output: R', refined rule
Method:

if $OvSubSpace$ is not empty
  $R' =$ **addOvFeature(**R**)**;
else
  $R' = R$;

return R';

Table 5. RefineWithUP-UN-Ov algorithm

Algorithm: RefineWithUP-UN-Ov

Input: R, a rule that needs refinement
Output: R', refined rule
Method:

if $UPSubSpace$ is not empty
  $R' =$ **addUPFeature(**R**)**;
else if $UNSubSpace$ is not empty
  $R' =$ **addUNFeature(**R**)**;
else if $OvSubSpace$ is not empty
  $R' =$ **addOvFeature(**R**)**;
else
  $R' = R$;

return R';

Table 6. RefineWithUN-UP-Ov algorithm

Algorithm: RefineWithUN-UP-Ov

Input: R, a rule that needs refinement
Output: R', refined rule
Method:

if $UNSubSpace$ is not empty
  $R' =$ **addUNFeature(**R**)**;
else if $UPSubSpace$ is not empty
  $R' =$ **addUPFeature(**R**)**;
else if $OvSubSpace$ is not empty
  $R' =$ **addOvFeature(**R**)**;
else
  $R' = R$;

return R';

Table 7. BestStrategy algorithm

Algorithm: BestStrategy

Input: R, rule that needs refinement
Output: R', refined rule
Method: $R_1 =$ **RefineWithUP**(R);
$R_2 =$ **RefineWithUN**(R);
$R_3 =$ **RefineWithOv**(R);
$R_4 =$ **RefineWithUP-UN-Ov**(R);
$R_5 =$ **RefineWithUN-UP-Ov**(R);

repeat
  $R_1 =$ **RefineWithUP**(R_1);
  $R_2 =$ **RefineWithUN**(R_2);
  $R_3 =$ **RefineWithOv**(R_3);
  $R_4 =$ **RefineWithUP-UN-Ov**(R_4);
  $R_5 =$ **RefineWithUN-UP-Ov**(R_5);
until R_1, R_2, R_3, R_4, R_5 do not need refinement

$R' =$ **ChooseBestRule**$(R_1, R_2, R_3, R_4, R_5)$;
return R';

every round of rule refinement, there is only one sub-space to consider. This may result in "forcing" a rule to be refined by one particular sub-space when using another sub-space may generate a better rule. To address this problem, a further two strategies were devised.

The first, BestPosRule, generates two refined versions of a rule, one version using a feature from the UP sub-space and the other using a feature from the Ov sub-space. The better of the two versions (again using Laplace estimation accuracy), is selected and is further refined in the same manner if required. Thus, this strategy uses two sub-spaces in conjunction during each rule refinement round and will only generate rules without negation.

Table 8. BestPosRule algorithm

Algorithm: BestPosRule

Input: R, a rule that needs refinement
Output: R', refined rule
Method:

repeat
    R_1 = **RefineWithUP**(R);
    R_2 = **RefineWithOv**(R);
    R = **ChooseBestRule**(R_1, R_2);
until R does not need refinement

$R' = R$;
return R';

The second, BestRule, is an extension of the BestPosRule strategy, where a third version of the rule to be refined is generated using a feature from the UN sub-space. Thus, this strategy uses all three sub-spaces for each round of rule refinement and may generate rules with negation. All three rule versions are compared and the best is selected based on the Laplace estimation accuracy. Intuitively, using more than one sub-space in conjunction in every round of rule refinement will generate better rules, as several versions of the rule to be refined are generated and the best selected. Table 8 and Table 9 shows the BestPosRule and the BestRule algorithms respectively. Table 10 summarizes all the strategies described in this section.

6 Datasets

Two well known text classification datasets, the 20 Newsgroups [15] and Reuters-21578 Distribution 1.0 [16] were used with respect to the evaluation reported in this paper. The 20 Newsgroups dataset is a collection of 19,997 documents, comprising news articles from 20 classes. There are 1,000 documents in each class

Table 9. BestRule algorithm

Algorithm: BestRule

Input: R, a rule that needs refinement
Output: R', refined rule
Method:

repeat
R_1 = **RefineWithUP**(R);
R_2 = **RefineWithOv**(R);
R_3 = **RefineWithUN**(R);
R = **ChooseBestRule**(R_1, R_2, R_3);
until R does not need refinement

$R' = R$;
return R';

Table 10. Summary of proposed rule refinement strategies

Strategy	Sub-space used
UP	UP only
UN	UN only
Ov	Ov only
UP-UN-Ov	either UP, UN or Ov in every round of rule refinement
UN-UP-Ov	either UP, UN or Ov in every round of rule refinement
BestStrategy	choose the best rule from the above five strategies
BestPosRule	UP and Ov in every round of rule refinement
BestRule	UP, UN and Ov in every round of rule refinement

with the exception of one class that contains 997 documents. In our experiments, this dataset was split into two non-overlapping datasets (hereafter, referred to as 20NG-A and 20NG-B), each comprising 10 classes. Therefore, 20NG-A has 10,000 documents, while 20NG-B has 9,997 documents. The classes in each split are shown in Table 11.

The Reuters-21578 Distribution 1.0 dataset is widely used to evaluate text classification tasks. It consists of 21,578 documents and 135 classes. In our experiments for single-labelled text classification, the preparation of this dataset followed the method suggested by Wang [19], where the top ten most populated classes were identified and multi-labelled/non-text documents were removed from each class. This leaves the dataset with only eight classes and 6,643 documents. Hereafter, this dataset is referred to as Reuters8. The classes and number of documents associated with each, in the Reuters8 dataset are shown in Table 12.

Table 11. Class names for 20NG-A and 20NG-B datasets

20NG-A	20NG-B
alt.atheism	comp.graphics
comp.sys.ibm.pc.hardware	comp.os.ms-windows.misc
comp.windows.x	comp.sys.mac.hardware
misc.forsale	rec.autos
rec.motorcycles	rec.sport.hockey
rec.sport.baseball	sci.crypt
sci.electronics	sci.space
sci.med	soc.religion.christian
talk.politics.mideast	talk.politics.guns
talk.religion.misc	talk.politics.misc

Table 12. Statistics for Reuters8 dataset

Classes	Num. of docs.
acq	2108
crude	444
earn	2736
grain	108
interest	216
money-fx	432
ship	174
trade	425

7 Evaluation Results

The experiments that were conducted compared the use of our proposed rule learning mechanism and rule refinement strategies with that of JRip, NaiveBayes and Sequential Minimal Optimization (SMO) from the Waikato Environment for Knowledge Analysis (WEKA) machine learning workbench [9]. χ^2 with a reduction factor of 0.9 was used as a dimensionality reduction method. Our rule learning mechanism is denoted as RL with the identifier for the different rule refinement strategies used appended. Table 13 reports the micro-averaged F1-measure from experiments using ten-fold cross validation, the overall average for each method, and the standard deviation (STDN) with respect to the overall average for the 20NG-A, 20NG-B and Reuters8 datasets. The top two best results in each case are shown in bold.

Comparison of the rule refinement strategies demonstrated that RL + BestRule is the best method. This is because RL + BestRule has the advantage of choosing from all three sub-spaces in every round of rule refinement. RL + UN (rule refinement using only negative features) seems to be the worst method. It is suggested that this is because of the structure of the rules produced by this strategy, where repeated rule refinement results in rules of the

Table 13. Micro-averaged F1-measure, overall average F1-measure, and standard deviation (STDN) for the 20NG-A, 20NG-B and Reuters8 dataset (top two best results shown in **bold**)

Method/Dataset	20NG-A	20NG-B	Reuters8	Average	STND
RL + UP	85.9	86.6	88.8	87.2	**1.5**
RL + UN	81.8	84.7	81.4	82.6	1.8
RL + Ov	84.8	85.8	81.8	84.1	2.1
RL + UP-UN-Ov	85.9	86.6	88.8	87.1	**1.5**
RL + UN-UP-Ov	85.6	86.8	91.5	88.0	3.1
RL + BestStrategy	85.4	86.6	**92.4**	88.1	3.7
RL + BestPosRule	**86.0**	86.4	89.1	87.2	1.7
RL + BestRule	**86.6**	**87.4**	**92.4**	**88.8**	3.1
JRip	76.0	80.8	91.8	82.9	8.1
NaiveBayes	63.5	65.6	80.4	69.8	9.2
SMO	84.8	**89.2**	**94.8**	**89.6**	5.0

form $a \wedge \neg b \wedge \neg c \wedge \neg d \wedge \neg e \Rightarrow x$. If a dataset contains many similar documents across all classes, where most documents include a and not b, c, d and e, then rules of this form would result in many misclassifications. RL + UP and RL + UP-UN-Ov produced the same results in all three datasets, suggesting that on no occasion was the UP sub-space empty. Thus, using UP or UP-UN-Ov would result in the same output. In all three datasets, RL + BestRule outperformed RL + BestPosRule. Overall, the results demonstrated that the methods that generate rules with negation perform better than the methods that generate rules without negation. Thus, rules with negation do indeed improve the effectiveness of text classification.

As expected, SMO produced good classification results, as support vector machine have been shown to be one of the best methods for text classification. SMO gave the best classification results in the 20NG-B and Reuters8 datasets, followed by RL + BestRule. In the 20NG-A dataset, RL + BestRule produced the best classification results. In addition, our rule learning mechanism with a number of strategies seemed to outperform SMO in the 20NG-A dataset. The best overall average F1-measure was obtained using SMO, however, with a relatively high standard deviation. Much better standard deviations were obtained using the RL algorithms indicating an element of consistency of operation across the datasets. In all cases, the performance of NaiveBayes was relatively poor. JRip did worse than our rule learning mechanism and SMO in the 20NG-A and 20NG-B datasets, but did better than NaiveBayes. In the Reuters8 dataset, JRip performed much better than a number of our rule refinement strategies, although RL + BestRule and RL + BestStrategy still did slightly better than JRip.

8 Conclusion and Future Work

In this paper, we described an investigation into refinement strategies for inducing rules with negated features. We have proposed a rule learning mechanism, based on the covering algorithm, that uses a number of strategies for rule refinement. The proposed strategies were founded on a division of the search space into three different sub-spaces: UP, UN and Ov. These strategies enable the learning of rules with and without negation. The reported evaluation was directed at determining the effectiveness of rules with negated features for text classification. Other methods for text classification, such as JRip, NaiveBayes and SMO were also used in the comparison. The best performing rule learning mechanisms was found to be RL + BestRule, which was shown to be better than JRip and NaiveBayes and competitive with SMO. The results indicate that by including negation in IRL, a better text classifier can be built. Future work will look into extending our research from using single keywords to the use of phrases. The use of phrases within the text representation vector is motivated by the potential benefit of preserving semantic information that is not present in the use of single keywords. Furthermore, we intend to investigate the use of negated phrases, as well as, the combination of keywords and phrases and their negation in text classification.

References

1. 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)
2. 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)
3. Baralis, E., Garza, P.: Associative text categorization exploiting negated words. In: Proceedings of the ACM Symposium on Applied Computing, pp.530-535 (2006)
4. Brunk, C., Pazzani, M.: Noise-tolerant relational concept learning algorithms. In: Proceedings of the 8th International Workshop on Machine Learning, Morgan Kaufmann (1991)
5. Cohen, W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning (ICML), pp. 115-123, Morgan Kaufmann (1995)
6. Crestani, F.: Learning strategies for an adaptive information retrieval system using neural networks. In: Proceedings of the IEEE International Conference on Neural Networks (1993)
7. Fürnkranz, J., Widmer, G.: Incremental reduced error pruning. In: Proceedings of the 11th International Conference on Machine Learning (ICML), Morgan Kaufmann (1994)
8. Galavotti, L., Sebastiani, F., Simi, M.: Experiments on the use of feature selection and negative evidence in automated text categorization. In: Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries, pp. 59-68 (2000)

9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H.: The WEKA data mining software: An update. In: SIGKDD Explorations **11** 10-18 (2009)
10. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2006)
11. Hayes, P. J., Weinstein, S. P.: CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In: Proceedings of the 2nd Conference on Innovative Applications of Artificial Intelligence (IAAI), pp. 49-66, AAAI Press (1990)
12. Holmes, G., Trigg, L.: A diagnostic tool for tree based supervised classification learning algorithms. In: Proceedings of the 6th International Conference on Neural Information Processing (ICONIP), pp. 514-519 (1999)
13. 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)
14. Johnson, D. E., Oles, F. J., Zhang, T., Goetz, T.: A decision-tree-based symbolic rule induction system for text categorization. In: The IBM Systems Journal, Special Issue on AI **41** 428-437 (2002)
15. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th International Conference on Machine Learning, pp. 331-339 (1995)
16. 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)
17. McCallum, A., Nigam, K.: A comparison of event model for naive Bayes text classification. In: Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, pp. 41-48 (1998)
18. 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)
19. Wang, Y. J.: Language-independent pre-processing of large documentbases for text classifcation. PhD thesis (2007)
20. Weiss, S. M., Indurkhya, N.: Optimized rule induction. In: IEEE Expert: Intelligent Systems and Their Applications **8** 61-69 (1993)
21. Wu, Z., Zhang, C., Zhang, S.: Mining both positive and negative association rules. In: Proceedings of the 19th International Conference on Machine Learning, pp. 658-665 (2002)
22. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval, pp. 42-49 (1999)
23. Zheng, Z., Srihari, R.: Optimally combining positive and negative features for text categorization. In: Proceedings of the International Conference on Machine Learning (ICML), Workshop on Learning from Imbalanced Datasets II (2003)