Using Negation and Phrases in Inducing Rules for Text Classification

Stephanie Chua, Frans Coenen, Grant Malcolm, and Matías Fernando García-Constantino

Abstract An investigation into the use of negation in Inductive Rule Learning (IRL) for text classification is described. The use of negated features in the IRL process has been shown to improve effectiveness of classification. However, although in the case of small datasets it is perfectly feasible to include the potential negation of all possible features as part of the feature space, this is not possible for datasets that include large numbers of features such as those used in text mining applications. Instead a process whereby features to be negated can be identified dynamically is required. Such a process is described in the paper and compared with established techniques (JRip, NaiveBayes, Sequential Minimal Optimization (SMO), OlexGreedy). The work is also directed at an approach to text classification based on a "bag of phrases" representation; the motivation here being that a phrase contains semantic information that is not present in single keyword. In addition, a given text corpus typically contains many more key-phrase features than keyword features, therefore, providing more potential features to be negated.

Stephanie Chua
Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, L69 3BX Liverpool, UK, e-mail: s.chua@liverpool.ac.uk

Frans Coenen
Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, L69 3BX Liverpool, UK, e-mail: coenen@liverpool.ac.uk

Grant Malcolm
Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, L69 3BX Liverpool, UK, e-mail: grant@liverpool.ac.uk

Matías Fernando García Constantino
Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, L69 3BX Liverpool, UK, e-mail: mattgc@liverpool.ac.uk

1 Introduction

Text mining is a well established component within the domain of Knowledge Discovery in Data (KDD) and especially data mining. One element of text mining is text classification where we wish to categorize documents according to a classifier generated using a training set. Many techniques have been proposed whereby the desired text classifier can be generated. Among the more popular techniques are the *k*-nearest neighbour (*k*-NN) [19], support vector machines (SVMs) [9], probabilistic Bayesian models [14, 19] and decision trees [8, 10] . The technique which is the focus of this paper is inductive rule learning (IRL) [1, 5]. The particular focus of the work described is IRL processes for text classification that incorporate an ability to dynamically include negated features in the rule learning process. The motivation here is twofold. Firstly, the inclusion of negated features in the IRL process can typically improve the quality of the resulting classifier. Indeed we can contrive text classification scenarios which can only be resolved by including negated features in the rule generation process (this was demonstrated in [4]). Secondly, the dynamic identification of candidate features that can be negated is seen as desirable as we do not wish to generate the complete set of potential negations a priori. In the case of datasets that have a small number of features, it is of course entirely feasible, and therefore justified, to include all potential feature negations as part of the "input"; however this is not justified in the case of datasets with very large numbers of features. The latter type of dataset is exemplified by the document collections to which text classification is typically applied. Such collections are typically represented, for text mining purposes, using the "bag of words" or "bag of phrases" representations. Document collections typically feature large numbers of keywords and even larger numbers of key-phrases. Consequently, the work described here is directed at the bag of phrases representation because this representation is likely to exhibit a greater number of potential features to be negated than in the case of the keyword representation. Thus, the objective of this paper is to evaluate the effectiveness of our proposed IRL mechanism against other machine learning techniques for text classification. In addition, the significance of using negation in IRL will be investigated, as well as, experimenting with the bag of phrases representation. An interesting point to evaluate include whether negated phrases can be more effective in text classification.

The potential inclusion of negated features in the IRL text classification process raises two issues. The first is the nature of the mechanism whereby we can identify the most appropriate negated features (without including all negated versions of the entire feature set). The second is the nature of the rule refinement strategies required to generate rules with and without negation (without using a fixed template as adopted in the case of some alternative approaches [15, 16]). A rule learning mechanism to include negation and a number of rule refinement strategies for addressing these issues are proposed and evaluated.

The rest of this paper is organized as follows. Section 2 describes some related work on IRL. Section 3 discusses our proposed mechanism for inductive rule learning with negation. Section 3.1 discusses the identification of negated features, and

Section 3.2 details the different rule refinement strategies used in our IRL approach. Section 4 discusses phrase extraction methods to extract phrases. The experimental setup is described in Section 5 and the results in Section 6. Section 7 concludes the paper.

2 Related Work

As noted in the previous section, many text classification techniques have been proposed. The technique at which the work described in this paper is directed is IRL. As in the case of some of the other techniques identified above, IRL offers the advantage that it is easily interpretable by human analysts. Many different IRL systems have been applied to the text classification problem. However, the focus in this paper is on systems that are capable of generating rules with negation. Examples of such systems include the Olex suite of systems [15, 16], and RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [5].

The Olex suite was developed by Rullo et al. and is founded on the idea of using a fixed template that allows only one positive feature and zero or more negative features to generate rules. The suite includes Olex Greedy and OlexGA. OlexGreedy, as the name suggests, uses a "greedy", single stage, rule learning process [15]. One of the disadvantages of OlexGreedy, highlighted by the authors, is that the template approach is not able to express co-occurrences based on feature dependencies. Rullo et al. attempted to overcome this disadvantage by using conjunction of terms (coterms). However, the authors again reported that rules that were generated using the improved version could not share common features in the antecedent. Hence, the authors proposed OlexGA [16], which uses a genetic algorithm to induce a rule-based classifier. This version overcame the problems associated with OlexGreedy. However, the generated rules still adhere to the fixed template of "one positive feature , zero or more negative feature(s)". A criticism of Olex is that the use of such templates is somewhat restrictive. Our IRL system proposes a number of rule refinement strategies that impose no restrictions on the number of positive or negated features.

RIPPER is an IRL system which uses the covering algorithm to learn rules whereby, when a rule is generated, the examples "covered" by the rule are removed from the training set (the process then repeats until all examples are covered). RIPPER generates rules by greedily adding features to a rule until the rule achieves a 100% accuracy. This process tries every possible value of each feature and chooses the one with the highest information gain. Following this rule building phase is a rule pruning phase, whereby the generated rule is pruned using a pruning metric. On the surface, it does not look like RIPPER includes any mechanism for explicitly generating rules with negation. However, in the case of binary-valued features, a feature-value of zero (0) is interpreted as a negated feature (the absence of a feature). The approach proposed in this paper also uses the covering algorithm; however, for reasons presented in Section 1, the search space does not include all possible negations of features from the feature set; instead these are identified as required.

3 Inductive Rule Learning with Negation

The proposed rule learning mechanism aims to improve the effectiveness of classifiers, comprising a small numbers of rules, by using both positive and negated features, while maintaining the simplicity and effectiveness of the covering algorithm. In the covering algorithm, rules are learned sequentially one at a time based on the training dataset. The documents "covered" by a rule learnt are then removed and the process is repeated until there are no more uncovered documents in the training set or there are no more unused features in the feature set. Rule refinement is a significant element of this approach. Suppose we have a rule $F \Rightarrow x$; in general, such a rule may cover both *positive* and *negative* documents: positive documents are documents in the training set that are correctly classified, while negative documents are those that are incorrectly classified. Rule refinement is used to obtain a more specialized rule $F \wedge l \Rightarrow x$. To prevent overfitting, or learning rules that are too precise, some stopping conditions to rule refinement were adopted. These conditions stop the rule refinement; (i) when a rule no longer covers negative documents, (ii) when the feature search space is empty or (iii) when the previous rule learnt has a higher or equal accuracy to that of the current rule learnt. Generating rules without negation is straightforward: we take l to be a conjunction of features that occur together in positive documents. However, generating rules with negation requires the identification of the feature to be negated. This will be discussed in Section 3.1. Our rule learning mechanism encompasses a number of strategies for rule refinement and these are discussed in Section 3.2.

3.1 Identifying Features

The discriminating 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 select the most discriminating features with respect to a specific class.

We distinguish two strategies for feature selection: local and global. In local feature selection, ordered features that are local to a specific class are selected for learning. 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 value is used to order and select features. In our experiments, despite a rigorous reduction factor of 0.9 (using only 10% of the features), global feature selection methods are still computationally expensive. We therefore focus on the local feature selection method.

In our proposed mechanism, during rule refinement, an appropriate feature is selected from the *local search space* of the rule. The search space contains features from both the positive and negative documents that are covered by the rule. Accordingly, we divide the search space into the following three sub-spaces.

1. **Unique Positive** (UP). Features that appear only in positive documents: we call these *unique positive* features.
2. **Unique Negative** (UN). Features that appear only in negative documents: we call these *unique negative* features.
3. **Overlap** (Ov). Features that are found in both positive and negative documents: we call these *overlap* features.

This division allows for the effective and efficient identification of positive or negated features to be used when refining rules. Note that for a given rule, the UP, UN and Ov sub-spaces may be empty, as the existence of these features depends upon the content of the documents covered by the rule.

When refining a rule, a feature from either the UP, UN or Ov sub-spaces can be selected to be added to the rule. If a UP or Ov feature is selected, it is simply added to the rule, and is not negated. If a UN feature is selected, then its negated form is added to the rule. When refining a rule with a UP or UN feature, we select the feature with the highest document frequency, i.e. the feature that occurs in the most covered documents. 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, we select the feature with the highest document frequency difference (i.e. positive document frequency minus negative document frequency). This is because an Ov feature occurs in both positive and negative documents and the feature that appears in the most positive documents and least negative documents will result in a refined rule that serves to maximise the number of positive documents.

3.2 Rule Refinement Strategies

There are a number of possible strategies for rule-refinement using the three sub-spaces, UP, UN and Ov. Here, we focus on eight of these. The first three strategies use only a single sub-space, from which they take their names: UP, UN and Ov. Table 1 shows a simple example of how the UP, UN and Ov strategies work.

Given that a sub-space may be empty, the UP, UN and Ov strategies may lead to refinement being prematurely halted in the absence of any features to be added to a rule. Two further strategies have been 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 are labelled in the order that the sub-spaces are considered. Thus, UP-UN-Ov entails the use of UP features first; if the UP sub-space is empty, then UN features will be considered instead, and then the Ov features if the UN sub-space is also empty. The UN-UP-Ov strategy works in a similar manner, only inter-

Table 1 Example of rule refinement with UP, UN and Ov strategies

Feature set for class x = {bike, ride, harley, seat, motorcycles, honda}
Initial rule learnt = $bike \Rightarrow x$
The rule covers three documents (two positive documents and one negative document)

Doc 1 labelled class x = {bike, ride, motorcycles}
Doc 2 labelled class x = {seat, harley, bike, ride}
Doc 3 labelled class y = {bike, ride, honda}

Identify UP, UN and Ov features
UP feature(s) = {motorcycles, seat, harley}
UN feature(s) = {honda}
Ov feature(s) = {ride}

Strategies for rule refinement
Refine with UP = $bike \wedge motorcycle \Rightarrow x$
Refine with UN = $bike \wedge \neg honda \Rightarrow x$
Refine with Ov = $bike \wedge ride \Rightarrow x$

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 features from that sub-space will be used for rule refinement. This means that the UP-UN-Ov strategy may produce the same results as the UP strategy, and similarly UN-UP-Ov may produce the same results as the UN strategy. For each rule to be refined, each of these five strategies may result in different rules. Our sixth strategy, BestStrategy, chooses the best rule (using accuracy with Laplace estimation) from the results of the first five strategies.

Each of the first five strategies refines a rule by selecting a feature from a particular sub-space; this refined rule is then further refined (using the same strategy) until some termination condition is met (e.g., the rule covers only positive documents, or further refinement produces a rule that is less accurate). Each of these strategies therefore corresponds to a depth-first search. A more exhaustive search through the possible rules is provided by our final two strategies. The first, BestPosRule, refines a rule by creating two versions of the original rule; one version by selecting a feature from the UP sub-space and another by selecting a feature from the Ov sub-space. Each of these rules is further refined in the same manner until the refined version is less accurate than the previous version. The rule with the best Laplace accuracy is then selected as the rule to be added to the ruleset. This strategy makes use of two sub-spaces during each refinement step and will only generate rules without negation. The second strategy, BestRule, is an extension of BestPosRule, where a third version of the rule to be refined is generated by selecting a feature from the UN sub-space. Thus, this strategy uses all three sub-spaces at each refinement step and may generate rules with negation. Rule refinement works in the same manner as in BestPosRule, but with an additional version where a feature from the UN sub-space is added. Again, the rule with the best Laplace accuracy will be the one added to the ruleset. Table 2 summarizes all the strategies described in this section.

Table 2 Summary of proposed rule refinement strategies

Strategy	Description	Sample rules
UP	Add a UP feature to refine a rule	$a \wedge b \Rightarrow x$
UN	Add a UN feature to refine a rule	$a \wedge \neg c \Rightarrow x$
Ov	Add an Ov feature to refine a rule	$a \wedge b \wedge d \Rightarrow x$
UP-UN-Ov	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	$a \wedge b \Rightarrow x$
UN-UP-Ov	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	$a \wedge b \wedge \neg c \Rightarrow x$
BestStrategy	Choose the best rule from the five rules generated by each UP, UN, Ov, UP-UN-Ov and UN-UP-Ov	$a \wedge b \wedge d \Rightarrow x$
BestPosRule	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	$a \wedge b \wedge d \wedge e \Rightarrow x$
BestRule	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	$a \wedge b \wedge \neg c \wedge \neg f \Rightarrow x$

4 The Bag of Phrases Representation

The use of the “bag of phrases” representation is motivated by the potential benefit of preserving semantic information that is not present in the “bag of words” representation. There are various methods that may be adopted to identify phrases for the bag of phrases representation. These methods tend to fall into two categories: linguistic phrase extraction and statistical phrase extraction. The former is based on linguistic patterns while the latter is based on statistical patterns.

Much previous work has reported on the use of phrases in text classification, albeit with mixed results. [6] investigated the use of linguistic phrases with both a naive bayes classifier (RAINBOW) and a rule-based classifier (RIPPER) and found that phrase features can improve classification at the expense of coverage. In [17], noun phrases and key phrases were extracted and used in RIPPER for text classification. The use of noun phrases was found to be only slightly better than the use of key words, while the use of key phrases was found to be slightly worse. In general, the authors reported no significant benefit from using phrases and concluded that more complex natural language processing methods were needed to identify them. In [2] phrases were extracted using a statistical word association based grammar, and an improvement over the use of the bag of words representation was reported using a naive bayes classifier. An n-gram word extractor was used in [13] to extract frequent phrases for classifying research paper abstracts using various classifiers; experiments showed that the bag of phrases representation was better than the bag of words representation for their dataset. [3] investigated the use of phrases for email classification and found that using phrases of size two gave the best classification results. However, none of the above investigated the use of negated phrases with respect to IRL in text classification. For the work described here, two phrase extraction mechanisms were adopted.

The first approach was founded on n-gram extraction and operated as follows:

1. Preprocess the dataset by removing stop words, numbers, emails and symbols.
2. Extract n-grams from the preprocessed dataset.
3. Sort the extracted n-grams from each class in descending order according to their chi-square values.
4. Select the top 10% of the extracted n-grams from each class to be used as features for representation.

The experiments reported later in this paper extracted three different kinds of n-grams: 1-gram (which is essentially single keywords), 2-grams and 3-grams.

The second approach was a variation on the n-gram extraction approach:

1. Preprocess the dataset by removing numbers, emails and symbols. Stop words are not removed.
2. Extract all single keywords in the dataset (not including stop words).
3. Sort the extracted single keywords from each class in descending order according to their chi-square values.
4. Select the top 10% of the single keywords from each class and store in features list.
5. Based on the selected single keywords, extract phrases from the dataset that contain at least one keyword from the features list.
6. Sort the extracted phrases from each class in descending order according to their chi-square values.
7. Select the top 10% of the extracted phrases from each class to be used as features for representation.

This approach extracts sequences of words from the dataset that still had stop words in them. Each phrase that was extracted consisted of at least one single keyword. The experiments reported in the next section reported the use of a two-word (Phrase-2) and a three-word (Phrase-3) based phrase extraction.

5 Experimental Setup

The experiments that were conducted compared the use of our proposed rule learning mechanism and rule refinement strategies with that of JRip, NaiveBayes (NB) and Sequential Minimal Optimization (SMO) from the Waikato Environment for Knowledge Analysis (WEKA) machine learning workbench [7]. In addition, an OlexGreedy plug-in to WEKA [16] was also compared. χ^2 with a reduction factor of 0.9 was used as a dimensionality reduction method. Both the n-gram and phrase extraction methods described in the previous section were considered.

Two well known text classification datasets, the 20 Newsgroups [11] and Reuters-21578 Distribution 1.0 [12] were used for the evaluation. 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 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 (20NG-A has 10,000 documents and 20NG-B 9,997 documents). This dataset was split only for computational efficiency reasons as reported in Wang [18], by taking 10 classes for 20NG-A and the remaining 10 classes for 10NG-B. Therefore, 20NG-A and 20NG-B should be viewed as two separate datasets and the results should be considered in this context.

The Reuters-21578 Distribution 1.0 dataset is widely used in text classification. 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 [18], where the top ten most populated classes were identified and multi-labelled/non-text documents were removed from each class. This resulted in a dataset with only eight classes and 6,643 documents. Hereafter, this dataset is referred to as Reuters8.

Table 3 shows the number of features used with respect to each class in each dataset. The number of features used is the top 10% of the potential set of features ordered using χ^2 that can be used to describe a class. The number of features increases from 1-gram to 3-grams and similarly, less phrases are extracted for Phrase-2 as compared to Phrase-3.

6 Evaluation

This section details the evaluation of the results obtained from the experiments conducted. Our rule learning mechanism is denoted as RL with the identifier for the different rule refinement strategies used appended. The micro-averaged F1-measure from experiments using ten-fold cross validation are reported.

Table 4 gives the classification results for the 20NG-A dataset. The RL average was computed for ease of comparison with the other machine learning methods. In the RL mechanism, the strategies that generated rules with negation produced the best results using the 1-gram and Phrase-2 representations, while strategies that generated rules without negation came in top for all but 1-gram representation. Comparison of the n-gram approach when using the RL mechanism shows that when using the 2-grams representation the best results are produced, while the worst results were generated using the 3-grams representation. RL+BestStrategy produced slightly better results than all the other RL strategies when a 1-gram representation was used. It was also better than JRip, NaiveBayes and OlexGreedy but was slightly worse than SMO. For the 2-grams, 3-grams, Phrase-2 and Phrase-3 representations, the RL mechanism produced the top two best results as compared to JRip, NaiveBayes, SMO and OlexGreedy. When using JRip, NaiveBayes, SMO and OlexGreedy, the

Table 3 10% of all the total features extracted for each class in each dataset

Dataset/Classes	1-gram	2-grams	3-grams	Phrase-2	Phrase-3
20NG-A					
rec.motorcycles	1205	4666	5197	1631	3247
talk.religion.misc	1597	7743	8836	3269	7334
sci.electronics	1208	5231	5824	1914	3740
alt.atheism	1442	7522	8628	3252	7530
misc.forsale	1150	4256	4908	2003	3749
sci.med	1837	8408	9524	3250	6553
talk.politics.mideast	1922	11433	13270	5613	12494
comp.sys.ibm.pc.hardware	1069	4929	5892	2404	5281
rec.sport.baseball	1086	5439	6369	2650	5507
comp.windows.x	1713	8377	9934	4260	8769
20NG-B					
comp.graphics	1395	6638	7723	2938	5919
comp.sys.mac.hardware	1062	4685	5421	2036	4371
rec.sport-hockey	1250	6738	8295	3560	7208
sci.crypt	1539	8000	9333	3399	7487
sci.space	1629	8287	9487	3198	6453
talk.politics.guns	1676	8282	9536	3354	7417
comp.os.ms-windows.misc	2709	10945	13001	6427	13864
rec.autos	1231	5318	5986	1905	3909
talk.politics.misc	1769	10312	11937	4090	9350
soc.religion.christian	1640	9868	11344	4756	11386
Reuters8					
acq	1283	9236	12724	5153	12579
crude	630	3451	4397	1694	3825
earn	1040	5607	8132	3298	8074
grain	263	820	925	257	485
interest	340	1473	1830	688	1542
money-fx	526	2885	3632	1494	3440
ship	365	1164	1310	448	851
trade	597	3488	4451	1780	4210

best results were produced using the 1-gram (keyword only) representation and the worst using the 3-grams representation.

The performance for the RL mechanism in the case of the 20NG-B dataset in Table 5 showed that the strategies that generated rules with negation came in the top two for all the representations, while strategies that generated rules without negation came top for all the representations with exception of the 1-gram representation. The 2-gram representation was better than 1-gram and 3-grams for the RL mechanism. Similar to the 20NG-A dataset, when using JRip, NaiveBayes, SMO and OlexGreedy, the best results were produced when using the 1-gram representation and the worst using the 3-grams representation. The RL+BestRule strategy was slightly better than the other RL strategies, as well as the JRip, NaiveBayes and OlexGreedy for the 1-gram representation; but was worse than the SMO. The RL mechanism again came into the top two when the 2-grams, 3-grams, Phrase-2 and

Table 4 Micro-averaged F1-measure for the 20NG-A dataset using both n-grams and phrases representation (top two best results shown in **bold**)

Method/Rep	1-gram	2-grams	3-grams	Phrase-2	Phrase-3
RL + UP	0.800	0.828	0.786	**0.920**	**0.873**
RL + UN	0.810	0.832	0.793	0.898	0.859
RL + Ov	0.803	**0.836**	**0.796**	0.894	0.859
RL + UP-UN-Ov	0.800	0.826	0.783	**0.920**	**0.873**
RL + UN-UP-Ov	0.810	0.832	0.788	0.901	0.862
RL + BestStrategy	**0.830**	0.833	0.791	**0.911**	0.864
RL + BestPosRule	0.824	**0.837**	**0.794**	0.907	**0.866**
RL + BestRule	0.821	0.831	0.789	0.910	0.864
RL Average	0.812	0.832	0.790	0.908	0.865
JRip	0.760	0.665	0.612	0.785	0.665
NaiveBayes	0.636	0.603	0.480	0.704	0.587
SMO	**0.849**	0.814	0.759	0.905	0.853
OlexGreedy	0.824	0.729	0.580	0.862	0.714

Phrase-3 representations were used, as compared to JRip, NaiveBayes, SMO and OlexGreedy.

Table 5 Micro-averaged F1-measure for the 20NG-B dataset using both n-grams and phrases representation (top two best results shown in **bold**)

Method/Rep	1-gram	2-grams	3-grams	Phrase-2	Phrase-3
RL + UP	0.844	**0.873**	0.827	**0.933**	**0.891**
RL + UN	0.825	0.862	**0.830**	0.911	0.881
RL + Ov	0.824	0.866	**0.831**	0.908	0.880
RL + UP-UN-Ov	0.844	**0.873**	0.826	**0.933**	**0.891**
RL + UN-UP-Ov	0.823	0.862	0.828	0.914	0.882
RL + BestStrategy	0.861	**0.869**	0.829	**0.920**	**0.884**
RL + BestPosRule	0.858	**0.869**	0.829	0.917	0.882
RL + BestRule	**0.862**	**0.869**	**0.830**	0.919	0.883
RL Average	0.843	0.868	0.829	0.919	0.884
JRip	0.808	0.754	0.694	0.844	0.746
NaiveBayes	0.656	0.654	0.540	0.734	0.604
SMO	**0.892**	0.858	0.800	0.895	0.873
OlexGreedy	0.845	0.780	0.619	0.890	0.758

The results for the Reuters8 dataset in Table 6 showed a slightly different trend than that for 20NG-A and 20NG-B. In the RL mechanism, the strategies that generated rules with negation produced the best top two results for 2-grams and 3-grams while strategies that generated rules without negation came in the top two for Phrase-2 and Phrase-3 and top in 3-grams. Again, 2-grams was still the best representation for the RL mechanism. However, 3-grams seemed to be slightly better for some of the strategies as compared to 1-gram. While JRip, SMO and OlexGreedy showed decreasing classification results in the order of 1-gram to 3-grams, Naive-Bayes had the best results when 2-grams was used, as opposed to 1-gram and 3-grams. The RL mechanism produced the best top two results for all representations except for 1-gram. In the 1-gram representation, the top two results were obtained using SMO and JRip. In fact, SMO had the best results for all the representations except for 3-grams.

Table 6 Micro-averaged F1-measure for the Reuters8 dataset using both n-grams and phrases representation (top two best results shown in **bold**)

Method/Rep	1-gram	2-grams	3-grams	Phrase-2	Phrase-3
RL + UP	0.822	0.879	0.851	**0.929**	**0.908**
RL + UN	0.842	0.873	0.862	0.908	0.887
RL + Ov	0.860	0.871	**0.867**	0.898	0.890
RL + UP-UN-Ov	0.822	0.879	0.851	**0.929**	**0.908**
RL + UN-UP-Ov	0.848	0.871	0.857	0.908	0.887
RL + BestStrategy	0.877	0.884	0.861	0.922	0.906
RL + BestPosRule	0.882	0.885	0.859	0.923	0.904
RL + BestRule	0.822	**0.887**	**0.863**	0.923	0.907
RL Average	0.847	0.879	0.859	0.918	0.900
JRip	**0.896**	0.844	0.735	0.907	0.854
NaiveBayes	0.775	0.802	0.698	0.843	0.799
SMO	**0.932**	**0.911**	0.840	**0.953**	**0.916**
OlexGreedy	0.883	0.875	0.787	0.915	0.875

The results obtained from the experiments suggested that when the RL mechanism was used to learn rules, the use of the phrase representations was beneficial with respect to text classification, particularly phrases of size two. This stemmed from the fact that, while the 1-gram representation was good enough as a representation for text classification, the rich nature of natural language text provided the use of phrases with the advantage of preserving semantic information that was not present in single keywords. However, three words appearing in sequence were likely to be occurring less frequently and too specific, and thus not appropriate for text classification. 2-grams in general could occur more frequently and serve to segregate two distinct classes when this could not be achieved using 1-gram. This however did not hold true for JRip, NaiveBayes, SMO and OlexGreedy where decreasing effectiveness was recorded when using 1-gram to 3-grams. All the techniques compared also showed that Phrase-2 was better than Phrase-3, strengthening the argument that any phrase longer than two was not effective for classification.

In the 20NG-A and 20NG-B dataset, the best RL strategy using the 1-gram representation was that which generated rules with negation and was competitive with SMO. In representations longer than one, strategies that generated rules without negation were slightly better in the 20NG-A and 20NG-B datasets. This suggests that the use of negated features is more effective when single keyword representation is used and less effective when the phrase representation is used. Single keywords can be quite common across different classes and thus, the use of negated features which are unique to other classes to learn rules that exclude documents from other classes seems to be effective. However, when phrases are used in the representation, the use of negation becomes “redundant”, due to the fact that a phrase itself can be unique enough to differentiate documents from other classes. A different scenario is depicted in the Reuters8 dataset though. For the Reuters8 dataset, the RL strategy which generated rules with negation was slightly better than the others when the 2-gram representations was adopted, but came in second for all the other representations. This could suggest that more common 2-grams occur across the different classes in the dataset.

As expected, SMO produced good classification results, as support vector machines have been shown to be one of the best techniques for text classification. It was the best technique for the 1-gram representation, but was outperformed by the RL mechanism for all the other representations for the 20NG-A and 20NG-B datasets. It was again the best technique for the Reuters8 dataset with respect to all representations except 3-grams. NaiveBayes consistently delivered the worst performance with respect to all datasets and all representations. The RL mechanism outperformed both JRip and OlexGreedy in all cases except for the 1-gram representation in the Reuters8 dataset where they were closely competitive.

7 Conclusion

An investigation into IRL with negation and phrases has been described. We have proposed an IRL mechanism, based on the covering algorithm, that includes a number of strategies for rule refinement. These strategies were devised based on the division of the search space into three different sub-spaces: UP, UN and Ov. A number of these strategies were designed to learn rules with negation. Experiments were carried out to evaluate the effectiveness of our IRL mechanism against that of other machine learning techniques. Interestingly, the evaluation showed that our IRL mechanism outperformed all the other machine learning techniques that were compared and was competitive with SMO. The experiments also aimed at investigating the effectiveness of rules with negation, as well as the bag of phrases representations for text classification. It was found that rules with negation were more effective when the single keyword representation was used and less prominent when the phrase representation was used. The use of phrases of size two was found to be beneficial for text classification while phrases longer than two seemed to be too unique to be useful.

References

1. 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)
2. Bakus, J., Kamel, M.: Document classification using phrases. In: Caelli, T. and Amin, A. and Duin, R. and de Ridder, D. and Kamel, M. (eds.): Structural, Syntactic, and Statistical Pattern Recognition, Lecture Notes in Computer Science, vol. 2396. Springer Berlin/Heidelberg, pp. 341-354 (2002)
3. Chang, M., Poon, C. K.: Using phrases as features in email classification. In: Journal of Systems and Software, Elsevier Science Inc., **82**, pp. 1036-1045 (2009)
4. Chua, S., Coenen, F, Malcolm, G.: Classification Inductive Rule Learning with Negated Features. In: Proceedings of the 6th International Conference on Advanced Data Mining and Applications (ADMA'10), Part 1, Springer LNAI, pp. 125-136 (2010)
5. Cohen, W.: Fast effective rule induction. In: Proceedings of the 12th Int. Conf. on Machine Learning (ICML), pp. 115-123, Morgan Kaufmann (1995)
6. Fürnkranz, J., Mitchell, T., Riloff, E.: A case study in using linguistic phrases for text categorization on the WWW. In: Working Notes of the AAAI/ICML Workshop on Learning for Text Categorization, AAAI Press, pp. 5-12 (1998)
7. 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)
8. Holmes, G., Trigg, L.: A diagnostic tool for tree based supervised classification learning algorithms. In: Proceedings of the 6th Int. Conf. on Neural Information Processing (ICONIP), pp. 514-519 (1999)
9. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the 10th European Conf. on Machine Learning (ECML), pp. 137-142 (1998)
10. 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)
11. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th Int. Conf. on Machine Learning, pp. 331-339 (1995)
12. 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)
13. Li, Z., Li, P., Wei, W., Liu, H., He, J., Liu, T., Du, X.: AutoPCS: A phrase-based text categorization system for similar texts. In: Li, Q., Feng, L., Pei, J., Wang, S., Zhou, X., Zhu, Q.-M. (eds.): Advances in Data and Web Management, Lecture Notes in Computer Science, vol. 5446. Springer Berlin/Heidelberg, pp. 369-380 (2009)
14. 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)
15. 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)
16. Rullo, P., Policicchio, V., Cumbo, C., Iiritano, S.: Olex: Effective rule learning for text categorization. In: Transaction on Knowledge and Data Engineering, **21:8** 1118-1132 (2009)
17. Scott, S., Matwin, S.: Feature engineering for text classification. In: Proceedings of the 16th Int. Conf. on Machine Learning (ICML), pp. 379-388 (1999)
18. Wang, Y. J.: Language-independent pre-processing of large documentbases for text classifcation. PhD thesis (2007)
19. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proceedings of the 22nd ACM Int. Conf. on Research and Development in Information Retrieval, pp. 42-49 (1999)