Feature Selection and Generalisation for Retrieval of Textual Cases

Nirmalie Wiratunga[1], Ivan Koychev[2], and Stewart Massie[1]

[1] School of Computing,
[2] Smart Web Technologies Centre,
The Robert Gordon University,
Aberdeen AB25 1HG, Scotland, UK
{nw|ik|sm}@comp.rgu.ac.uk

Abstract. Textual CBR systems solve problems by reusing experiences that are in textual form. Knowledge-rich comparison of textual cases remains an important challenge for these systems. However mapping text data into a structured case representation requires a significant knowledge engineering effort. In this paper we look at automated acquisition of the case indexing vocabulary as a two step process involving feature selection followed by feature generalisation. Boosted decision stumps are employed as a means to select features that are predictive and relatively orthogonal. Association rule induction is employed to capture feature co-occurrence patterns. Generalised features are constructed by applying these rules. Essentially, rules preserve implicit semantic relationships between features and applying them has the desired effect of bringing together cases that would have otherwise been overlooked during case retrieval. Experiments with four textual data sets show significant improvement in retrieval accuracy whenever generalised features are used. The results further suggest that boosted decision stumps with generalised features to be a promising combination.

1 Introduction

Past problem solving experiences captured in textual form present an interesting challenge to CBR system development. This is because experiences in unstructured form containing free text must first be mapped into structured cases before they can be meaningfully compared and reused for future problem solving. Textual CBR (TCBR) involves reuse of experiences that are in text form [14]. Unlike Information Retrieval approaches TCBR aims to develop case representation mechanisms that can better support knowledge-rich comparison of cases.

TCBR systems often access a variety of knowledge sources (e.g. domain specific thesauri, natural language parsers etc.) to establish an indexing vocabulary [5]. The general aim is to facilitate structured case representation and enhance retrieval. In this paper we investigate how introspective learning can be employed to automate the acquisition of the case indexing vocabulary [13]. We present techniques that are generally applicable when textual experiences are pre-classified according to the types of problems they solve. Essentially we shall exploit implicit knowledge already existing in text documents to discover keywords that on their own or as a set in combination with others, are predictive of the problem class. The case indexing vocabulary will constitute

just these selected keywords and so this process can be viewed as dimension reduction or feature selection.

Feature selection techniques employed by machine learning algorithms for supervised learning tasks such as classification are known to successfully improve accuracy, efficiency and comprehension of learned concepts [12]. Typically these techniques have been applied in problem domains consisting of structured cases. They have also been employed by CBR systems to identify relevant features for building an index for case retrieval [11]. A feature selection technique can be categorised as either being a filter or a wrapper approach. The wrapper approach uses feedback from the final learning algorithm to guide the search for the set of features. Generally this feedback ensures selection of a good set of features tailored for the learning algorithm but has the disadvantage of being time consuming because feedback involves learner accuracy ascertained from cross-validation runs. Filters are seen as data pre-processors and generally do not require feedback from the final learner. As a result they tend to be faster, scaling better to large datasets. Selection techniques presented in this paper fall under filter approaches which are particularly suited to processing of medium to large text collections.

In classification problems a *good* feature is one that is predictive of the problem class on its own or in combination with other features. Selection according to the performance of a combination of features is particularly useful for text data because there is often the need to identify similar meaning words that are used interchangeably (synonyms) and the same word being used with different meaning (polysemies). In both situations similar cases can be overlooked during retrieval if these semantic relationships are ignored. This paper introduces a novel feature selection technique that discovers and preserves semantic relationships in the case representation as part of the selection process. Boosted decision stumps are used for feature selection and semantic relationships are captured using association rule induction.

Section 2 describes the commonly used information gain based feature selection technique which is then used by the boosted feature selection technique in Section 3. The Apriori association rule learner is discussed in Section 4 and is employed as a means to capture semantic relationships between features. In Section 5, induced rules are utilised to form a generalised document representation and in doing so introduces novel ways of combining it with feature selection. Experimental results are reported on four textual classification tasks in Section 6. An overview of case representation and indexing issues in textual CBR research and how techniques presented in this paper relate to existing ones are discussed in Section 7, followed by conclusions in Section 8.

2 Feature Selection with Information Gain

We first introduce the notation used in this paper to assist presentation of the different feature selection techniques. Let $\mathcal{D}$ be the set of all labelled documents, $\mathcal{W}$ the set of all features which are essentially words. A document d is a pair $(\vec{x}, y)$, where $\vec{x}$ = $(x_1, \ldots, x_{|\mathcal{W}|})$ is a binary valued feature vector corresponding to the presence or absence of words in $\mathcal{W}$; and y is d's class label [18]. The experiments in this paper use binary class domains so y is either 0 (negative class) or 1 (positive class). Let $\mathcal{S}$ be the training subset containing labelled documents $\{d_1, \ldots, d_n\}$.

The main aim of feature selection is to reduce $|\mathcal{W}|$ to a smaller feature subset size m by selecting features ranked according to some goodness criteria. The selected m features then form a new binary-valued feature vector $\vec{x}'$ and a corresponding reduced word vocabulary set $\mathcal{W}'$, where $\mathcal{W}' \subset \mathcal{W}$ and $|\mathcal{W}'| \ll |\mathcal{W}|$. The new representation of document d with $\mathcal{W}'$ is a pair $(\vec{x}', y)$.

A feature's discriminatory power is a useful gauge of its goodness and is commonly ascertained using the information gain (IG) score ([17], [16]).

$$IG(X,Y) = \sum_{x \in 0,1} \sum_{y \in 0,1} P(X = x, Y = y).log_2 \frac{P(X = x, Y = y)}{P(X = x).P(Y = y)}$$

Here the probabilities are estimated from $\mathcal{S}$ using m-estimates [15]. The information gain based ranking and selection of features is the base line algorithm used in this paper and we will refer to it as BASE (Figure 1).

m = feature subset size
BASE
  Foreach $w_i \in \mathcal{W}$
    calculate IG score using $\mathcal{S}$
  sort $\mathcal{W}$ in decreasing order of IG scores
  $\mathcal{W}' = \{w_1, \ldots, w_m\}$
  Return $\mathcal{W}'$

Fig. 1. Feature selection with IG based ranking.

A feature goodness score like IG reflects a feature's ability to discriminate between classes. A possible shortfall with BASE is that selected features although having high scores may exercise their discriminatory power in similar ways. Consider documents from two mailing lists about computer hardware, one list containing messages about solving PC problems and the other dedicated to Apple Macs. An example of the top ranked words might be: "centris", "quadra", "eisa", "bus", "client", "server" etc. Here both "centris" and "quadra" are likely to suggest a hidden concept such as machine type. Similarly "eisa" and "bus" are likely to co-occur in similar documents an possibly relate to an implicit concept like internal architecture, while "client" and "server" are also features that can be viewed as belonging to a further implicit concept such as process communication. Ideally we would like to explicate these semantic relationships but firstly we need to ensure that as many of the hidden concepts are captured by at least a single representative discriminatory feature. This means that if we were restricted to select just three out of the six words a useful selection might be: "quadra", "eisa" and "server" to cover each of the hidden concepts; instead of just the top three "centris", "quadra" and "eisa". What this example is highlighting is that selecting just the top ranked features with BASE can result in a feature set that is not particularly representative of hidden concepts thereby having a detrimental effect on case comparison. In the

following section we combine IG based feature selection with boosting as a first step towards dealing with this problem.

3 Feature Selection with Boosted Decision Stumps

Boosting is known to improve the performance of learning algorithms particularly with tasks that exhibit varying degrees of difficulty [9]. The general idea of boosting is to iteratively generate several (weak) learners, with each learner biased by the training set error in the previous iteration or trial. Each learner works hard at solving training instances that were incorrectly classified in previous iterations. This is achieved by associating weights with instances in the training set and updating these weights at each trial. Weights of instances correctly solved by the most recent learner are decreased, and this has the effect of increasing weights of incorrectly classified instances. It means that at the next trial the learner is forced to work harder at solving these difficult instances. In order to classify a new test instance, the votes of each learner are combined to form a majority vote. Each vote is typically weighted by learner accuracy because it makes sense to trust those learners that have a higher accuracy on the training set.

An interesting approach to feature selection is to use boosting with a one-level decision tree, known as a decision stump, as the learning algorithm ([6], [8]). Constructing such a learner involves selecting a single feature, based on its ability to discriminate between classes [10]. For this purpose decision stumps are typically formed from features with high information gain. An example of two decision stumps from the binary classed computer hardware domain appear in Figure 2. Here a "+" denotes documents from the Apple mailing list and "-" the PC mailing list. With the "centris" stump the left leaf is formed by documents in which "centris" is present and the right leaf contains documents where it is absent. Predicting the class of a test document using this decision stump involves traversing the left or right branch leading to a leaf depending on the presence or absence of "centris" and labelling the document with the majority class at that leaf. Similar explanations hold for the stump having "bus" as the splitting feature. The stump error on the training set (err) is the percentage of the number of minority class documents in both branches.

Since a decision stump partitions the domain based on the values of a single feature, the set of stumps generated with boosting form the set of selected features. Therefore with m boosted iterations a set of m features are selected and these form the reduced feature subset $\mathcal{W}'$. The BOOST feature selection technique is shown in Figure 3. At each boosted iteration the feature with highest IG is selected forming the stump for the training set $\mathcal{S}$. Initially all n documents are assigned the same weight of $1/n$. With each trial these weights are updated so that the weights of correctly classified examples are reduced according to the error of the stumps. In practice once weights are updated, they need to be re-normalised so that their sum remains one. The impact of updated weights will be reflected in the IG scores where the prior and conditional probabilities are calculated on weighted documents, and this in turn will influence the feature selected in the next iteration when forming the stump. The boosting mechanism adopted here is similar to AdaBoost.M1 [9], the only difference being that updating of document weights is based on the error of the committee of stumps learned thus far, instead of the error

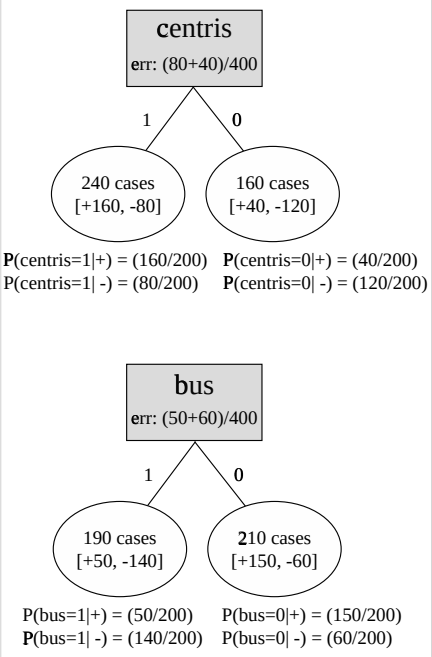


Fig. 2. Decision stumps.

$\mathcal{W}' = \emptyset$
stumps = $\emptyset$
max-trial = m
trial = 1

BOOST
Foreach $d_i \in \{d_i, \ldots, d_n\}$
initialise d_i's weight to 1 / n
Repeat
F = highest ranked features from BASE
F = F $\setminus$ $\mathcal{W}'$
w_j = highest of F
stumps = stumps $\cup$ DecisionStump(w_j,$\mathcal{S}$)
$\mathcal{W}' = \mathcal{W}' \cup w_j$
err = error rate of stumps on $\mathcal{S}$
Foreach $d_i \in \mathcal{S}$
if d_i is correctly classified by stumps
update d_i's weight = weight * (err/1-err)
re-normalise all weights
++trial
Until (trial = max-trial)
Return $\mathcal{W}'$

Fig. 3. Feature selection with boosted stumps.

of the most recent decision stump. With initial stumps containing features with higher IG scores the committee approach to updating document weights enables stumps from earlier iterations to exert a greater influence on feature selection.

Features that are discriminatory in similar ways have less opportunity to be selected with BOOST. However, with most tasks, information about which features co-occur with selected features can provide useful knowledge for case similarity, particularly in the presence of hidden concepts. In the next section we use an association rule learner to identify co-occurring features for selected features. A generalised feature space formed by applying these learned rules to selected features provides a richer case representation which in turn will enrich case comparison.

4 Feature Generalisation with Association Rule Induction

Apriori [1] is a well known association rule induction algorithm introduced for the market-basket analysis domain where one wishes to find regularities in people's shopping behaviour. It generates rules of the form H $\leftarrow$ B, where the rule body B is a conjunction of items, and the rule head H is a single item. Association rules are discovered in two stages. Firstly Apriori identifies sets of items that frequently co-occur, i.e. above a given minimum threshold. It then generates rules from these itemsets ensuring frequency and accuracy are above minimum thresholds.

4.1 Rule Generation and Selection

```
+r1:centri<- print (6.5%, 17.2%, 0.3%)
+r2:centri<- card  (6.6%, 25.4%, 1.1%)
+r3:centri<- fpu   (5.5%, 24.5%, 0.8%)
+r4:centri<- iisi  (7.7%, 14.5%, 0.1%)
+r5:centri<- simm  (9.0%, 16.3%, 0.3%)
+r6:centri<- quadra(10.8%, 24.0%, 1.5%)
+r7:centri<- lc    (9.0%, 16.3%, 0.3%)

-r1:bus <- local  (7.7%, 46.4%, 3.0%)
-r2:bus <- standard  (10.3%, 31.5%, 1.2%)
-r3:bus <- window  (13.6%, 29.5%, 1.2%)
-r4:bus <- id  (20.5%, 28.3%, 1.7%)
-r5:bus <- drive  (29.6%, 31.7%, 4.8%)
-r6:bus <- id local  (9.0%, 42.0%, 2.7%)
-r7:bus <- drive local  (10.3%, 37.0%, 2.1%)
```

Fig. 4. Example list of rules from the hardware domain.

An obvious analogy exists between frequently occurring itemsets in shopping transactions and frequently occurring words in a set of documents. This means that rules can be used to predict the presence of the head feature given that all the features in the body are present in the document. This means that a case satisfying the body even when the head feature is absent, will be considered closer to other cases that actually have the head feature present. Figure 4 lists two sets of rules generated for the hardware mailing list domain. The first rule set corresponds to rules generated with "centris" as the rule head and the other set with "bus" as the head. The class of documents from which these rules were induced are indicated by the rule prefix. This is important because co-occurrences of features are a signature of a particular class of documents.

In order to tie in a set of rules to a class it is necessary to constrain rule generation so that a rule's body contains features that are predictive of the same class as the rule's head, and learning is restricted to documents from this class. The predictive class of features is estimated according to class conditional probabilities. Going back to Figure 2, if "centris" is to be used as the head feature of the rule then the higher conditional probability, $P(centris = 1|+)$ indicates that it is most likely to appear in documents from the positive class. If instead "bus" is the head feature then the higher conditional probability $P(bus = 1|-)$ suggests the negative class.

An informed rule selection strategy is necessary because Apriori typically will generate many rules [3]. The percentages in Figure 4 are the coverage, accuracy and information gain for each generated rule. Generally the first two measures are used by Apriori during rule generation to prune the search space. Here coverage (or frequency) is the percentage of documents in which a rule is applicable; and confidence (or accuracy) is the proportion of documents in which the rule prediction is correct. The third measures the gain in information due to the rule's body, and indicates how well the body is able to predict the presence or absence of the head feature. It is this measure that we have found most informative when selecting the K best rules from those gen-

erated. The three best rules predictive of each of the two head features (i.e. “centris”, “bus”) according to information gain are in bold.

4.2 Feature Generalisation

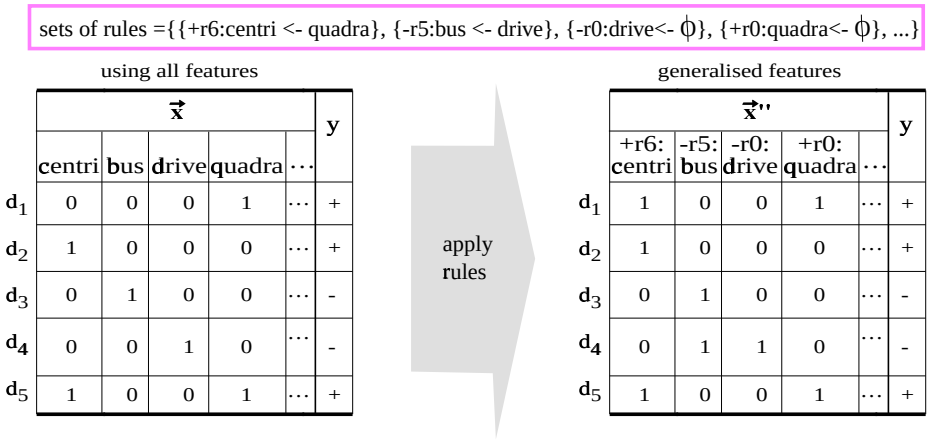
The objective of applying learned association rules is to improve case comparison by providing a more generalised case representation. Good generalisation will have the desired effect of bringing cases that are semantically related closer to each other that previously would have been incorrectly treated as being further apart. Association rules are able to capture implicit relationships (e.g. like synonyms) that exist between features. When these rules are applied they have the effect of squashing these features, which can be viewed as feature generalisation.

For a feature $w_i \in \mathcal{W}$, let $\mathcal{R}_i$ be the set of association rules induced with w_i as the head feature, where $r_{ij} : w_i \leftarrow B_j$. Here the rule body B_j is a conjunction of features from $\mathcal{W} \setminus \{w_i\}$ and when true implies the presence of the head feature w_i. Given a document’s initial representation $d = (\vec{x}, y)$ (i.e. using all features in $\mathcal{W}$), the generalised representation $d = (\vec{x}'', y)$ is obtained by applying $r_{ij} : x_i \leftarrow x_{i1} \wedge \ldots \wedge x_{in_j}$ where $x_{ik} \neq x_i$, giving;

$$x_i'' = \begin{cases} 1 & \text{if } x_i = 1 \\ 1 & \text{if } (\bigwedge_{k=1}^{n_j} x_{ik}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

All this means is that x_i'' is instantiated with value 1 if either the head of the rule or its body is true, and is 0 otherwise. Consequently, the generalised new document representation $\vec{x}''$ tends to be less sparse than $\vec{x}$, because 0 values are likely to have their values flipped to 1. Essentially $\vec{x}''$ remains a binary valued feature vector, whose values indicate the presence or absence of a feature w'', where $w'' \in \mathcal{W}''$, but $\mathcal{W}'' \not\subset \mathcal{W}$, since these features no longer correspond to presence or absence of single words.

Figure 5 illustrates how rules are used to generalise feature vectors. Here two forms of five trivial feature vectors are shown. The left table shows values for each vector using all the features in $\mathcal{W}$ = {“centri”, “bus”, “drive”, “quadra”,...}, with the y column showing the document class. The right table shows the effect of generalisation after the sets of rules are applied. For sake of simplicity we use only the single best rule from each of the rule sets $\{\mathcal{R}_{centri}, \mathcal{R}_{bus}, \mathcal{R}_{drive}, \mathcal{R}_{quadra} \ldots\}$; listed at the top of the figure. The first two rule sets contain a complete rule each: $\mathcal{R}_{centri} = \{+r6 : centri \leftarrow quadra\}$, $\mathcal{R}_{bus} = \{-r5 : bus \leftarrow drive\}$. So for example any rule from $\mathcal{R}_{centri}$ (e.g. $+r6 : centri$) is applied to the left table’s “centri” column on any document from the positive class, while rules from $\mathcal{R}_{bus}$ are applied to the “bus” column on any document from the negative class. The right table is the result of applying these rule sets. The other two rule sets: $\mathcal{R}_{drive}$ and $\mathcal{R}_{quadra}$ contain rules that have empty bodies. Such rules are not uncommon and indicate that Apriori was unable to find rules above specified minimum thresholds. Applying empty rules amounts to unchanged values, i.e. no generalisation takes place.



	$\vec{x}$					y
	centri	bus	drive	quadra	⋯	
d_1	0	0	0	1	⋯	+
d_2	1	0	0	0	⋯	+
d_3	0	1	0	0	⋯	-
d_4	0	0	1	0	⋯	-
d_5	1	0	0	1	⋯	+

	$\vec{x}''$					y
	+r6: centri	-r5: bus	-r0: drive	+r0: quadra	⋯	
d_1	1	0	0	1	⋯	+
d_2	1	0	0	0	⋯	+
d_3	0	1	0	0	⋯	-
d_4	0	1	1	0	⋯	-
d_5	1	0	0	1	⋯	+

Fig. 5. Example of generalisation with rules.

5 Combining Feature Selection with Generalisation

An obvious manner in which to perform generalisation is after feature selection. In Figure 6 BASEGEN does exactly this using BASE first to form $\mathcal{W}'$. It then uses $\mathcal{W}'$ as a handle on ruleset generation, where a ruleset $\mathcal{R}_i$ is generated for each selected feature $w'_i \in \mathcal{W}'$. This restricts the number of generated rule sets to m, so $|\mathcal{W}''| = |\mathcal{W}'|$. Here a rule $r_{ij} \in \mathcal{R}_i$ is of the form $r_{ij} : w'_i \leftarrow B_j$, where the rule body B_j is still a conjunction of features in $\mathcal{W} \setminus \{w'_i\}$, but the head now applies to a selected feature in $\mathcal{W}'$, where $\mathcal{W}' \subset \mathcal{W}$.

Interestingly we can also combine feature generalisation with boosted feature selection so that the boosted search for the best set of features is influenced at each iteration by the generalisation of the feature selected in the previous iteration. BOOSTGEN achieves this as shown in Figure 7. It calls $generalise$ before forming the decision stump, as a result the decision stump is formed by splitting the training set according to the new generalised feature.

Generalisation after feature selection is attractive because generated rules will contain rule bodies that bring in features from the larger feature pool $\mathcal{W}$. In this manner both BASEGEN and BOOSTGEN are able to link selected features from $\mathcal{W}'$ with other less frequently used features. This may be seen as supplementing selected features in $\mathcal{W}'$ with background knowledge from $\mathcal{W}$. Additionally BOOSTGEN's boosted feature selection will tend to discover generalised features that are less likely to have overlapping semantic relationships with other generalised features.

6 Evaluation

Feature selection and generalisation techniques enable the mapping of textual documents into structured cases with which the case base is formed. Different case representations are formed using the 4 algorithms presented in this paper:

$\mathcal{W}'' = \emptyset$; $\mathcal{W}' = \emptyset$
BASEGEN
call BASE to form $\mathcal{W}'$
Foreach $d_i \in \mathcal{S}$
Foreach $w_j \in \{\mathcal{W}' \cap \mathcal{W}\}$
x''_{ij}=generalise(x_{ij}, w_j)
w''_j=new generalised feature
$\mathcal{W}'' = \mathcal{W}'' \cup w''_j$
Return $\mathcal{W}''$

generalise(x, w)
$\mathcal{R}$ = select-rules(w)
apply each rule in $\mathcal{R}$
generalising x to x''
Return x''

select-rules(w)
$\mathcal{R}$ = rules with w as rule head
sort $\mathcal{R}$ decreasing order of rule IG
break ties with coverage
retain the best K in $\mathcal{R}$
Return $\mathcal{R}$

Fig. 6. Generalisation after feature selection.

$\mathcal{W}'' = \emptyset$; $\mathcal{W}' = \emptyset$; stumps = $\emptyset$
max-trial = m
trial = 1

BOOSTGEN
⋮
Repeat
F = highest ranked features from BASE
F = F $\setminus$ $\mathcal{W}'$
w_j = highest of F
$\mathcal{W}' = \mathcal{W}' \cup w_j$
Foreach $d_i \in \mathcal{S}$
x''_{ij} = generalise(x_{ij}, w_j)
w''_j = new generalised feature
stumps = stumps $\cup$ DecisionStump(w''_j,$\mathcal{S}$)
$\mathcal{W}'' = \mathcal{W}'' \cup w''_j$
err = error rate of stumps on $\mathcal{S}$
⋮
++trial
Until (trial = max-trial)
Return $\mathcal{W}''$

Fig. 7. Generalisation with boosted selection.

1. BASE, feature selection using the standard IG ranking (Figure 1);
2. BOOST, feature selection with boosting (Figure 3);
3. BASEGEN, generalisation after feature selection (Figure 6); and
4. BOOSTGEN, generalisation in combination with boosting (Figure 7)

The case retrieval performance using test set accuracy with 3 nearest neighbours is used to compare the above algorithms. A modified case similarity metric is used to refrain from treating the absence of words in the same way as the presence of words. This is because the presence of a word in documents is intuitively more important for measuring their similarity, than its absence. We accomplish this affect by weighting the similarity in non-present words by the inverse of the feature subset size. What this means is that as increasing number of features are used to represent documents, the influence of similarity due to the absence of similar words is reduced.

Textual cases were formed by pre-processing documents by firstly removing stop words (common words) and special characters such as quote marks, commas and full stops (except for "!", "@", "%", "$" because they have been found to be discriminative for some domains [17]). Remaining words are reduced to their stem using the Porter's algorithm. Essentially, $\mathcal{W}$ is formed by all word-stems ($|\mathcal{W}| \approx 8000$) remaining after document pre-processing. For our experiments we use pre-processed documents from the following text corpuses:

– **LingSpam** dataset has been formed to study the problem of spam. It contains 2893 email messages, of which 83% are non-spam messages related to linguistics, and rest are spam [17].
– **20 Newsgroups** dataset is a corpus of about 20,000 Usenet news postings into 20 different newsgroups. One thousand messages from each of the twenty newsgroups were chosen at random and partitioned by newsgroup name [15]. For our experiments we use three sub-corpuses, where the messages from two newsgroups are combined to form a binary classification as follows: Religion and Politics (RelPol); Apple Mac and PC Hardware (MacPc); and Space and Medical Science (SpcMed).

We created equal sized disjoint training and test sets, where each set contains 20% of documents randomly selected from the original corpus, preserving class distribution in the original corpus. For repeated trials, 15 such train test splits are formed. Significance is reported from a paired one tailed t-test with 99% confidence. The graphs show averaged accuracy on test set with increasing number of selected features.

6.1 Results

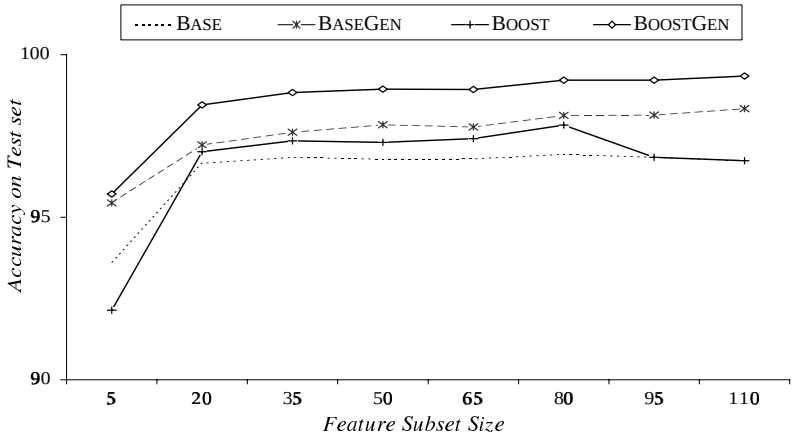


Fig. 8. Accuracy results for LingSpam.

The general behaviour of all four algorithms with the LingSpam corpus indicate an initial steep rise in accuracy (upto 20 features) after which there is hardly any improvement with increasing numbers of features (see Figure 8). The generalisation achieved with BASEGEN has resulted in a small but significant increase in accuracy over BASE, while BOOST has only managed a slight improvement. However, BOOSTGEN's generalisation combined with boosting has significantly outperformed the other algorithms, achieving the highest accuracy approaching 99%. The overall accuracy results suggest that this domain is relatively easy because BASE achieves 93.6% accuracy with only five features and improves this accuracy to over 97% with twenty features and above.

The reason for this is due to the nature of the LingSpam corpus, where there are a few very discriminatory features from non spam messages that are sufficient to differentiate spam messages.

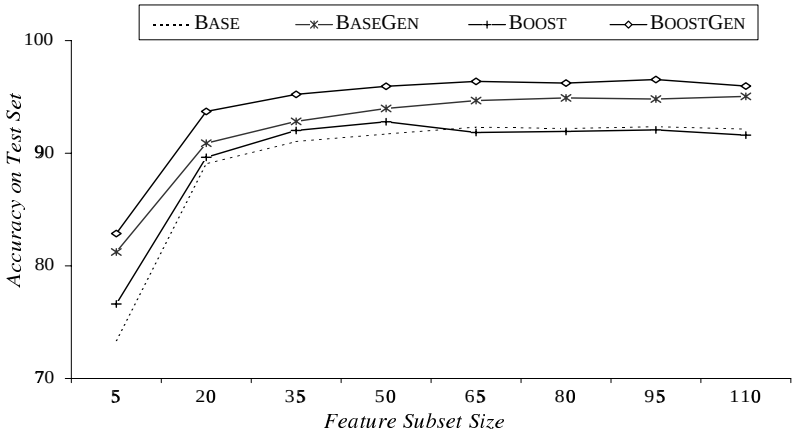


Fig. 9. Accuracy results for RelPol.

Figure 9 shows the results with the RelPol task. Compared to LingSpam the classification of documents in to Religion and Politics seems to present a harder task because overall accuracy is lower. BOOST results are comparable to BASE where boosted feature selection shows improved accuracy with relatively smaller feature subset sizes. As before algorithms employing generalisation (BASEGEN and BOOSTGEN) outperform those without generalisation (BASE and BOOST), with BOOSTGEN having significantly improved performance over all other algorithms (including BASEGEN).

The results from the MacPc classification task appear in Figure 10. This task is expected to be the hardest, because similar terminology (e.g. monitor, hard drive) can be used in reference to both PC and Apple Mac hardware. Additionally the same hardware problem can be applicable in both mailing lists resulting in cross posting of the same message. Although boosting on its own has not improved accuracy, boosting combined with generalisation (BOOSTGEN) is significantly better than all other algorithms including BASEGEN at all feature subset sizes. Interestingly the accuracies for algorithms using generalisation (BASEGEN and BOOSTGEN) continue to rise with increasing feature subset sizes. The poor performance of BOOST can be explained by the relatively low discriminatory power of features in this domain. In fact selecting the most discriminatory feature followed by boosting of incorrectly classified documents can be harmful, because updating of document weights prevents discovery of supportive features in subsequent boosted iterations.

A similar significant increase in classification accuracy with generalisation compared to without it is seen with the SpcMed domain (see Figure 11). Noticeably the overall winner here is BASEGEN having done significantly better than BOOSTGEN for the first time. Furthermore, boosting is not helpful and its performance is signifi-

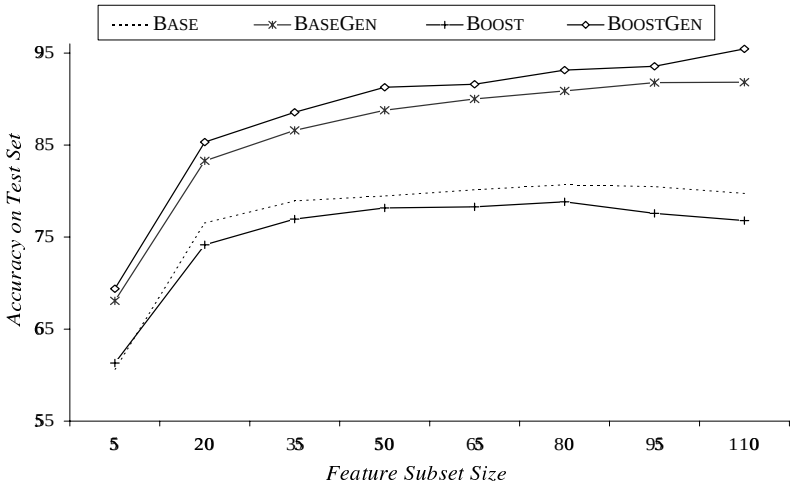


Fig. 10. Accuracy results for MacPc.

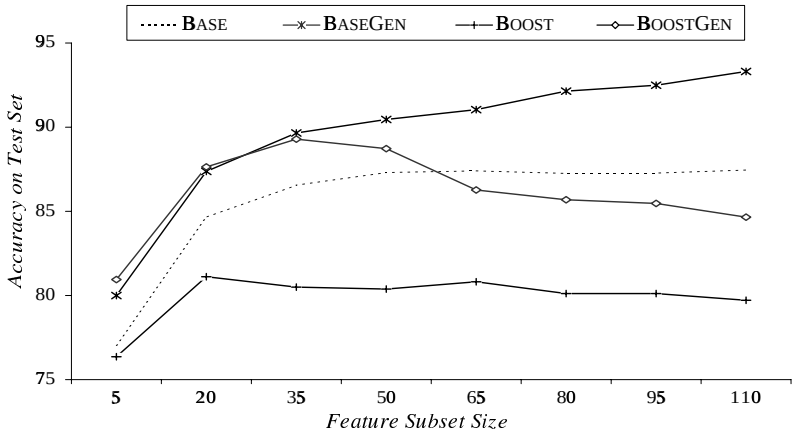


Fig. 11. Accuracy results for SpcMed.

cantly worse than BASE. Closer examination of BOOST's results indicate over-fitting behaviour, because the accuracy on training set is higher than that of BASE's accuracy on training set, but this gain is not reflected in test set accuracy. The generalisation used in BOOSTGEN maintains comparable performance to BASEGEN with up to 35 features, after which accuracy drops quickly as more features are used and over-fitting from boosting takes effect.

6.2 Evaluation Summary

The results from the significance tests are summarised in Table 1. The first two columns convey the gain with boosting (BOOST vs. BASE and BOOSTGEN vs. BASEGEN); and

Table 1. Results summary according to significance.

	Boosting		*Generalisation*	
	BOOST	BOOSTGEN	BASEGEN	BOOSTGEN
Data Set	vs. BASE	vs. BASEGEN	vs. BASE	vs. BOOST
LingSpam	no diff.	√	√	√
RelPol	no diff.	√	√	√
MacPc	×	√	√	√
SciMed	×	×	√	√

the other two the gain with generalisation (BASEGEN vs. BASE and BOOSTGEN vs. BOOST). Overall feature generalisation improves algorithm performance significantly It is worth noting that generalisation is able to continuously improve accuracy with increasing feature subset sizes with all domains, making it clearly more robust to overfitting. Generally boosting is not helpful on its own, but BOOSTGEN combining boosting with generalisation achieves significant improvement over all other algorithms in 3 out of 4 domains.

7 Related Work

Current practice in TCBR system development show that the indexing vocabulary and similarity knowledge containers are typically acquired manually [19]. This is not surprising because of the ambiguous nature of free text. Although NLP tools can be applied to analyse free text they are often too brittle partly because they tend to analyse text from a purely linguistic point of view. Instead a piecemeal approach involving increasing levels of knowledge intensive containers have been identified as the basis for TCBR system development [13]. Generally these levels are broadly seen as connected with the case representation vocabulary or the similarity measure. Tools such as stemming, stop word removal and domain specific dictionaries form less intensive knowledge levels and are mostly automated. Acquiring semantic relationships between words typically form higher knowledge levels and are harder to automate and remain an important challenge.

The difficulty with acquiring an appropriate indexing vocabulary and the need for structured case representation within the law domain is discussed in [4]. The SMILE system adopts a fine-grained sentence level class, whereby sentences are manually categorised into classes. It is interesting to note that although our approach does not explicitly assign classes at the sentence level, we also found it necessary to automatically link induced rules to applicable document classes. SMILE employs a decision tree based index scheme to partition the case base, but this is only possible after case sentences are manually marked-up (with words specified in a domain specific thesauri) to mitigate the synonym problem. We believe that our approach to feature generalisation with association rules helps automate the extraction of synonym relationships, provided that these relationships are already implicit in the textual case base.

Association rules have previously been used to reduce sparseness of initial user rating tables in collaborative recommendation [2]. Unlike traditional correlation based

approaches Apriori is able to capture statistics about co-occurring features efficiently because it exploits the fact that no superset of an infrequent itemset can be frequent. Work presented in this paper combines feature selection with rule induction providing a useful strategy to manage rule generation and selection. Additionally the boosting in our approach attempts to capture features that tend to be orthogonal and with which hidden concepts can be discovered by exploiting rules generated by Apriori.

The aims of feature generalisation discussed in this paper are similar to those of Latent Semantic Indexing (LSI); a popular dimension reduction technique for text data. It uses singular value decomposition to map the word based feature vector representation into a lower dimensional latent space of artificial features [6]. Recently LSI was also integrated with textual case retrieval, where case similarity is computed on the basis of the lower dimensional case representation [7]. Unlike LSI our approach to feature vector generalisation explicitly captures hidden semantic relationships by way of association rules, enabling easier interpretation of generalised features during case comparison. Still it will be intriguing to see how the feature selection and generalisation techniques introduced in this paper compare with LSI based case representation.

8 Conclusions

The idea of feature generalisation and combining this with feature selection to form structured cases for textual retrieval is a novel contribution of this paper. Feature generalisation helps tone down ambiguities that exist in free text by capturing semantic relationships and incorporating these in the case representation. This enables a much better comparison of cases.

The two main approaches presented in this paper are feature selection with boosting and feature generalisation with association rules. Essentially feature selection helps with identifying discriminatory features while feature generalisation captures semantic relationships. Overall case representation with generalisation significantly improved accuracy over algorithms without generalisation, and promises great potential for automated acquisition of both the indexing vocabulary and the similarity containers. The effect of boosting is mixed where on its own gives modest improvement or even harmful in some domains, where it is more prone to over-fitting. Further research is needed to understand the relationship between types of problem domains and boosting performance. However the best results in 3 of the 4 test domains were obtained by the combination of generalisation with boosting.

An interesting observation is that with feature selection and generalisation a more effective case retrieval is achieved even with a relatively small set of features. This is attractive because smaller vocabularies can effectively be used to build concise indices that are understandable and easier to interpret.

Acknowledgements

We thank Susan Craw, Rob Lothian and Dietrich Wettschereck for helpful discussions on this work.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.: Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, 307–327. AAAI/MIT Press (1995)
2. Alvarez, W., Ruiz, C.: Collaborative recommendation via adaptive association rule mining. In *Proceedings of the International Workshop on Web Mining for E-Commerce* (2000) 35–41
3. Borgelt, C., Kruse, R.: Induction of association rules: Apriori implementation. In *Proceedings of the 14th Conference on Computational Statistics* (2002)
4. Bruninghaus, S., Ashley, K.: Bootstrapping case base development with annotated case summaries. In *Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR'99* (1999) 59–73
5. Bruninghaus, S., Ashley, K.: The role of information extraction for textual CBR. In *Proceedings of the 4th International Conference on Case-Based Reasoning, ICCBR'01* (2001) 74–89
6. Cai, L., Hofmann, T.: Text categorisation by boosting automatically extracted concepts. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2003) 182–189
7. Chakraborti, S., Ambati, S., Balaraman, V., Khemani, D.: Integrating knowledge sources and acquiring vocabulary for textual CBR. In *Proceedings of the 8th UK-CBR workshop* (2003) 74–84
8. Das, S.: Filters, wrappers and a boosting based hybrid for feature selection. In *Proceedings of the 18th International Conference on Machine Learning*, Morgan Kaufmann (2001) 74–81
9. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning* (1996)
10. Iba, W., Langley, P.: Induction of one-level decision trees. In *Proceedings of the Ninth International Workshop on Machine Learning* (1992) 233–240
11. Jarmulak, J., Craw, S., Rowe, R.: Genetic algorithms to optimise CBR retrieval. In Enrico Blanzieri and Luigi Portinale, editors, *Proceedings of the 5th European Workshop on Case-Based Reasoning*, Trento, Italy, Springer-Verlag, Berlin (2000) 137–149
12. John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In *IML94* (1994) 121–129 Journal version in AIJ.
13. Lenz, M.: Defining knowledge layers for textual CBR. In *Proceedings of the 4th European Workshop on Case-Based Reasoning*, Dublin, Ireland, Springer Verlag (1998)
14. Lenz, M.: Knowledge sources for textual CBR applications. In *In Proceedings of the AAAI-98 Workshop on Textual Case-Based Reasoning*, Menlo Park, CA, AAAI Press (1998) 24–29
15. Mitchell, T.: *Machine Learning*. McGraw-Hill International (1997)
16. Pazzani, M. J., Muramatsu, J., Billsus, D.: Syskill and Webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR (1996) 54–61
17. Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C., Stamatopoulos, P.: A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6 (2003) 49–73
18. Salton, G., McGill, M. J.: An introduction to modern information retrieval. McGraw-Hill (1983)
19. Weber, R., Aha, D. W., Sandhu, N., Munoz-Avila, H.: A textual case-based reasoning framework for knowledge management applications. In *Proceedings of the 9th German Workshop on Case-Based Reasoning. Shaker Verlag.* (2001)