Maintenance by a Committee of Experts: The MACE Approach to Case-Base Maintenance

Lisa Cummins and Derek Bridge

Department of Computer Science,
University College Cork,
Ireland
{l.cummins,d.bridge}@cs.ucc.ie

Abstract. Case-base administrators face a choice of many maintenance algorithms. It is well-known that these algorithms have different biases that cause them to perform inconsistently over different datasets. In this paper, we demonstrate some of the biases of the most commonly-used maintenance algorithms. This motivates our new approach: maintenance by a committee of experts (MACE). We create composite algorithms that comprise more than one individual maintenance algorithm in the hope that the strengths of one algorithm will compensate for the weaknesses of another. In MACE, we combine algorithms in two ways: either we put them in sequence so that one runs after the other, or we allow them to run separately and then vote as to whether a case should be deleted or not. We define a grammar that describes how these composites are created. We perform experiments based on 27 diverse datasets. Our results show that the MACE approach allows us to define algorithms with different trade-offs between accuracy and the amount of deletion.

1 Introduction

In this paper we examine the most commonly-used case-base maintenance algorithms, and we present a new approach to maintenance, the MACE approach, which uses a committee of experts to make maintenance decisions.[1]

Case-base maintenance has the goal of restoring a degree of efficiency to the retrieval step of the CBR cycle by removing cases from the case-base, while, at the same time, preserving, or even enhancing, the accuracy of the system. The most common case-base maintenance algorithms are listed in Table 1.

As the Table shows, there are two types of case-base maintenance algorithm: those that delete noisy (or harmful) cases, and those that delete redundant cases. Noise reduction algorithms improve solution quality by removing cases that are considered to have a negative effect on system accuracy. Redundancy reduction algorithms improve system efficiency by removing cases which do not contribute to case-base competence.

[1] This material is based upon work supported by the Science Foundation Ireland under Grant Number 05/RFP/CMS0019.

Name used in this paper	Name in the literature	Description
Atomic noise reduction algorithms		
RENN	RENN	Repeated Edited Nearest Neighbour [15]
BBNR	BBNR	Blame-Based Noise Reduction [5]
Atomic redundancy reduction algorithms		
ICFR	—	Redundancy reduction phase of ICF [2]
RCR	—	Redundancy reduction phase of RC [10]
CRR	CRR	Conservative Redundancy Reduction [5]
Classic composite algorithms		
RENN→ICFR	ICF	Brighton & Mellish's Iterative Case Filtering [2]
RENN→RCR	RC	McKenna & Smyth's algorithm [10]
BBNR→CRR	CBE	Delany & Cunningham's Case-Base Editing algorithm [5]

Table 1. Atomic case-base maintenance algorithms, and classic composites

In practice, case-base maintenance algorithms are often composites, comprising a noise reduction phase followed by a redundancy reduction phase. For example, Brighton & Mellish's Iterative Case Filtering (ICF) algorithm comprises a RENN noise-filtering phase followed by their redundancy reduction phase [2].

The individual components of these composites have not always been tested individually, nor have they been tested in combination with other of the algorithms. For example, how does ICF's redundancy reduction phase perform if it is not preceded by RENN? How does it perform if it is preceded by BBNR instead of RENN? To answer questions like these, we need to separate the composites into their two constituent parts. This is why we have extracted the redundancy reduction phase of the composite algorithms, naming them in the Table, and treating them as separate algorithms. For example, ICFR is our designation for the redundancy reduction phase of ICF.

In the next section we will analyse these algorithms in greater detail. Section 3 presents the MACE approach to case-base maintenance. Section 4 presents our experimental methodology. Then Sections 5, 6, 7 and 8 present overall results, results concerning noise reduction algorithms, results concerning the effect of class boundary complexity, and results for the special case of spam, respectively.

2 Comparison of Existing Algorithms

2.1 Empirical comparisons

It is well-known that the composite algorithms in Table 1 perform differently on different datasets (see, e.g., [2]). The results in Table 2 from our own implementations of these algorithms exemplify this.[2] (Our experimental methodology is explained in detail later in this paper, Section 4.)

[2] Our implementations are publicly available as they are part of the open source jColibri framework, `http://gaia.fdi.ucm.es/projects/jcolibri/`.

	27 datasets		Breathalyser		Credit		Lenses	
Algorithm	Del (%)	Acc (%)	Del (%)	Acc (%)	Del (%)	Acc (%)	Del (%)	Acc (%)
RENN→ICFR	78.76	73.58	77.53	**74.00**	84.02	83.38	44.38	52.50
RENN→RCR	**88.75**	75.09	**87.66**	66.80	**87.84**	**86.38**	**86.25**	55.00
BBNR→CRR	55.31	**77.67**	61.95	71.20	55.90	83.62	68.13	**65.00**

Table 2. Results for existing algorithms with highest results highlighted

If we consider the results averaged over 27 datasets, we see that RENN→RCR is the most aggressive and BBNR→CRR is the most conservative. Perhaps surprisingly, RENN→RCR is only slightly behind BBNR→CRR in accuracy even though it deletes over 30% more. Also, even though RENN→ICFR deletes 10% less than RENN→RCR, it is less accurate. However, since these results are averaged over 27 datasets, they hide details about individual datasets. For example, although RENN→RCR beats RENN→ICFR in the average results, this is not necessarily the case for each dataset.

If we look at the results from some of the individual datasets we see that RENN→ICFR has highest accuracy on the Breathalyser dataset, RENN→RCR on the Credit dataset, and BBNR→CRR on the Lenses dataset. On all three datasets, RENN→RCR deletes most. RENN→ICFR deletes more than BBNR→CRR on two of the datasets but the reverse is true for the Lenses dataset.

All of this simply serves to confirm what is well-known: in case-base maintenance, there is no clear 'winning' algorithm. The differences in performance of the algorithms are caused by their having different biases.

2.2 An analysis of algorithm biases

Each of the different atomic maintenance algorithms targets different types of cases to remove. In this section, we attempt to further illustrate these biases by looking at some scenarios in detail.

The two atomic noise reduction algorithms define noise differently. RENN regards a case as noisy if it has a different class to the majority of its k nearest neighbours. After each case has been checked, RENN deletes all cases that are flagged as noisy. This is then repeated until no more cases are removed [15]. BBNR identifies and removes cases which cause other cases to be misclassified [5]. It first classifies each case in the case-base using its neighbours. It removes neighbours that cause misclassifications, provided their removal does not cause cases that were previously correctly classified to become misclassified. Figures 1 and 2 illustrate scenarios which reveal biases in RENN and BBNR.

If we take $k = 3$, in the situation shown in Figure 1, each case has one nearest neighbour of the same class as itself and two of the other class. This means that the majority of the nearest neighbours of each case are of a different class and so RENN will flag each for deletion, leaving this case-base empty after the first deletion pass is made. This problem does not occur with BBNR because c_4 does

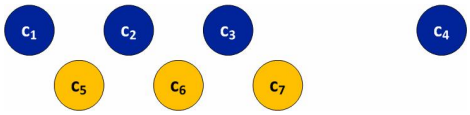

Fig. 1. RENN problem situation

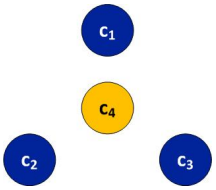

Fig. 2. BBNR problem situation

not cause any misclassification and so it will not be considered for deletion. It will be left as the only case in the case-base after BBNR is run.

The bias is the opposite in the scenario shown in Figure 2. Here, if we take $k = 3$, each of c_1, c_2 and c_3 contributes to the misclassification of c_4, but if any of them is deleted then the others would be misclassified, so BBNR will delete nothing. RENN will only delete c_4 because it is the only case with the majority of its k nearest neighbours of a different class.

In the case of the atomic redundancy reduction algorithms, all aim to remove cases that do not contribute to coverage, but they differ in how they decide which cases contribute. As a result they delete different types of cases. ICFR and CRR both aim to retain cases on the boundaries between classes because these cases are important for classification accuracy. ICFR removes cases that are solved by more cases than they themselves solve [2]. CRR removes cases that solve other cases [5]. It arranges the cases in the case-base in ascending order of how many cases they solve. It then adds each case c to a new case-base and removes from the original case-base any cases that c solves. RCR, however, aims to retain a case if it is surrounded by many cases of the same class, while deleting those that surround it [10]. It orders cases by descending relative coverage, which is a measure of the coverage contribution of each case in relation to how much it itself is covered. As each case is added to a new case-base, the cases that it solves are removed from the original case-base. Given these different biases, RCR will typically delete more cases than either ICF or CRR but this more aggressive deletion may result in lower accuracy in datasets with complex class boundaries.

It is apparent from both our experimental results and our brief analysis of biases that it is not the case that 'one size fits all' in case-base maintenance. The question naturally arises whether novel composites, that combine algorithms with different biases, can do better than the atomic algorithms and the classic composites. We explore this in the next section, in which we present the MACE approach, where maintenance is done by a committee of experts.

<*System*> ::= <*AtomicAlgorithm*> | <*Sequence*> | <*Committee*>
<*AtomicAlgorithm*> ::= <*AtomicNoiseReductionAlgorithm*> |
<*AtomicRedundancyReductionAlgorithm*>
<*AtomicNoiseReductionAlgorithm*> ::= 'RENN' | 'BBNR'
<*AtomicRedundancyReductionAlgorithm*> ::= 'ICFR' | 'RCR' | 'CRR'
<*Sequence*> ::= <*System*>'→'<*System*>
<*Committee*> ::= '{' <*System*> <*System*> <*System*>* '}' $^{\langle VotingRule \rangle}$
<*VotingRule*> ::= 'S' | 'M' | 'U'

Fig. 3. The MACE approach, defined by an EBNF grammar

3 Maintenance by a Committee of Experts (MACE)

In the MACE approach, we consider each atomic algorithm to be an expert that recommends cases for deletion. The MACE approach then defines different ways in which these atomic algorithms combine to form novel composite case-base maintenance algorithms. The easiest way to show how MACE forms these composites is by giving a grammar. This grammar is shown in EBNF in Figure 3.

The grammar's start symbol is <*System*>, which has three expansions:

<*System*> **::=** <*AtomicAlgorithm*> In the simplest case, a maintenance system comprises just one of the atomic algorithms. The atomic algorithms are here divided into the noise reduction algorithms (RENN, BBNR), and the redundancy reduction algorithms (ICFR, RCR, CRR).

<*System*> **::=** <*Sequence*> A composite maintenance system may put systems into sequence, denoted by writing an arrow between them. An example is a classic composite such as BBNR→CRR. When we write this, we mean that the algorithm comprises two phases, where the second (CRR) is executed after the first (BBNR). This rule allows us to create all of the classic composites but novel composites that have not previously been tested too, such as ICFR→BBNR. (The recursion in the grammar also allows the possibility of sequences that comprise more than two algorithms, although this is a degree of freedom that we shall not explore in this paper.)

<*System*> **::=** <*Committee*> Another way to create a composite is to form a committee (from which the MACE approach takes it name). A committee comprises a set of two or more systems, which we write between curly braces. Each system within a committee is executed, but cases are not deleted. If a member of a committee would ordinarily delete a case, we treat this instead as a vote for the deletion of that case. Committees must therefore also have a voting rule, which we write in superscript following the closing curly brace, which determines how votes are tallied. We explain the voting rules in the next paragraph. An example of a committee is $\{\text{BBNR, RENN}\}^{U}$ where each of BBNR and RENN separately proposes a set of cases to delete; and the committee deletes each case for which the voting is unanimous (U).

We define three voting rules for committees:

Single (S): If any member of the committee votes to delete case c, then the committee deletes c.

Majority (M): If more than half of the members of the committee vote to delete c, then the committee deletes c.

Unanimous (U): If all of the members of the committee vote to delete c, then the committee deletes c.

Single voting allows us to define committees that can aggressively delete large parts of a case-base, especially if the committees' constituents have very different biases. For example, $\{\text{BBNR, RENN}\}^S$ deletes all the cases that BBNR identifies as noisy, plus all the cases that RENN identifies as noisy. On the other hand, with unanimous voting, we can define very conservative committees. For example, if $\{\text{BBNR, RENN}\}^U$ deletes a case, we can be fairly confident that the case is noisy since both algorithms agree.

The real power of the grammar, however, comes from the mutual recursion in the rules. We will illustrate this with just three examples.

Since a sequence comprises two systems, and a system can be a committee, the grammar allows for sequences of committees. An example is $\{\text{BBNR, RENN}\}^S \rightarrow \{\text{ICFR, RCR, CRR}\}^U$, which first runs an aggressive noise reduction committee, followed by a very conservative redundancy reduction committee.

Similarly, since a committee comprises two or more systems, and a system can be a sequence, the grammar allows for committees of sequences. An example is $\{\text{RENN}\rightarrow\text{ICFR, RENN}\rightarrow\text{RCR, BBNR}\rightarrow\text{CRR}\}^M$, which uses majority voting of the three classic algorithms.

Finally, since a committee comprises two or more systems, and a system can be another committee, the grammar allows for committees with sub-committees. An example is $\{\{\text{RENN, BBNR}\}^U, \text{RCR}\}^S$, in which RCR votes alongside a noise sub-committee.

Related work. The idea of combining techniques with different biases is not new. In machine learning, ensembles classify new problems using each of the classifiers in the ensemble [6, 14]. Ensembles have similarly been been used in CBR [4, 9, 12]. Similarly, distributed CBR [13] deals with the use of multiple case-bases and how the system works in combining these.

The work in case-based ensembles and in distributed CBR tends to imply the use of multiple case-bases, with goals such as improved accuracy, efficiency, or personalisation. Brodley & Friedl also use multiple case-bases (in fact, multiple folds of the same case-base) in their approach to case-base maintenance [3]. They split the case-base and use a classifier that is trained on one part to classify the remaining part; they repeat this for each of several splits; and they then remove any cases that were misclassified. This use of multiple case-bases, however, is not a feature of the MACE approach: we are combining different *algorithms*. Closest to our work is arguably Wiratunga et al [17]. We are using a committee of maintenance experts, whereas they use a committee of adaptation experts.

Dataset Name	Cases	Features	Classes	Accuracy (%)
Balance	625	4	3	85.12
Breast Cancer Diagnostic	569	30	2	96.90
Breast Cancer Prognostic	198	33	2	71.58
Breathalyser	127	5	2	71.60
Credit Approval	690	15	2	86.92
Dermatology	366	34	6	97.75
Flags	194	28	8	52.89
Glass Identification	214	9	7	69.05
Haberman's Survival	306	3	2	69.51
Heart Disease Cleveland	303	14	5	53.22
Hepatitis	155	19	2	80.63
Ionosphere	351	33	2	86.71
Iris	150	4	3	97.00
Lenses	24	4	3	72.50
Lettings	756	5	2	84.97
Liver Disorders	345	6	2	64.20
Lung Cancer	32	56	3	48.00
Pima Indians Diabetes	768	8	2	70.78
Post-Operative Patient	90	8	3	64.71
Spam (5 datasets)	1000	700	2	95.55
Teaching Assistant Evaluation	151	5	3	55.33
Wine	178	13	3	96.67
Zoo	101	16	7	91.50
Average over 27 datasets	-	-	-	79.46

Table 3. Details of the datasets used in experiments

4 Experimental Methodology

Datasets. Table 3 lists the 27 datasets that we use to evaluate maintenance algorithms in this paper: 20 from the UCI repository [1]; the Breathalyser dataset [7]; the Lettings dataset [11]; and five email datasets [5]. We have datasets of varying sizes, with different numbers of attributes and different numbers of classes. The datasets also have varying amounts of noise and redundancy.

MACE algorithms. The MACE grammar defines an infinite set of maintenance algorithms. In our experiments, we put a number of restrictions on the sequences and committees that we created. We limited the sequences to ones that comprise either two atomic algorithms or one atomic algorithm and one committee, and we obviously ensured that a sequence contained distinct algorithms (e.g. BBNR→BBNR is excluded). We similarly excluded duplicates from committees, and kept the length to five or less. We allowed sub-committees, but not sub-sub-committees, and a committee that contained a sub-committee could only contain one other component (which was allowed to be an atomic al-

gorithm, a sequence or a sub-committee). With all of these restrictions in place, we created 307 algorithms from the grammar for experimentation.

Methodology. For evaluation, we performed repeated holdout on each of the datasets. Each dataset was divided randomly into three splits: a 60% training set, a 20% test set, and a final 20% which was required for evaluation of other systems in our research (not reported in this paper) and hence was discarded here. We created 10 different splits of the data.

We ran each algorithm on the training set and recorded the percentage of cases deleted. We also recorded the accuracy of the resulting case-base by using the test set as queries and recording the percentage correctly classified. We also recorded the accuracy before performing any maintenance to provide a benchmark figure. Table 3 contains these benchmark accuracies.

5 General Results

In this section, we compare overall performance, averaged over the 27 datasets. But there is an immediate problem: case-base maintenance is a multi-objective problem. We wish to optimise both the percentage of cases deleted, but also the accuracy of the final case-base. This is not possible: algorithms that do well on one of the criteria do not necessarily do well on the other. It comes as no surprise, for example, that our experimental results show that committees with many members and single voting delete many cases, but at a severe cost in accuracy; the opposite is the case with committees with few members and unanimous voting. Case-base administrators must strike a balance between accuracy and deletion. They need ways of seeing the trade-offs, so they can make informed decisions. We looked at two ways of presenting this.

5.1 Harmonic mean

We could present the arithmetic mean of the percentage of cases deleted and the case-base accuracy. But this can be misleading. The arithmetic mean in the case of an algorithm with very high accuracy and very low deletion will be similar to the arithmetic mean in the case of an algorithm with medium accuracy and deletion. To avoid this, we instead use the harmonic mean (Equation 1). The harmonic mean penalises large differences between two values so that a high mean is only produced if both individual values are high. In this way we can find algorithms which have a high value for both accuracy and deletion.

$$HarmonicMean(Acc, Del) = \frac{2 \times Acc \times Del}{Acc + Del} \tag{1}$$

Table 4 shows the algorithms with the top five harmonic means. We can see that these algorithms have high values for both accuracy and deletion.

These algorithms perform well when compared with the average accuracy when no deletion occurs (79.46%). We can see that the algorithm with the best

Algorithm	Deletion (%)	Accuracy (%)	Harmonic mean
{RENN, BBNR, RCR}^{S}	90.12	74.94	81.83
RENN→RCR	88.75	75.09	81.35
{RENN, RCR}^{S}	88.20	74.15	80.57
RCR→BBNR	89.67	72.63	80.26
{{RENN, BBNR}^{U}, RCR}^{S}	83.26	77.42	80.23

Table 4. Top five algorithms ordered by harmonic mean over 27 different datasets

accuracy, {{RENN, BBNR}^{U}, RCR}^{S}, only causes a 2% drop in accuracy while deleting 83.26%. This seems to be a reasonable compromise.

Additionally, we can see that novel MACE algorithms are performing well. In the top five, there are three committees, all of which are quite conservative. There is also one novel sequence, RCR→BBNR, and one classic, RENN→RCR. Interestingly, all five are some combination of RCR and a noise reduction algorithm. On its own, the atomic RCR algorithm has much lower deletion (76.67%) and therefore a much lower harmonic mean value (76.53). Our results show the strength of combining algorithms in sequences and committees: the weaknesses in RCR that cause the accuracy drop are compensated for by the noise algorithms.

A problem with this way of presenting the results, however, is that it does not give the case-base administrator much sense of what trade-offs can be made. For example, what does she do if she is not happy to see accuracy fall by 2%.

5.2 Pareto front

Another way to handle a multi-objective problem is to compute the Pareto front. The Pareto front contains algorithms that are not dominated by any other algorithms. We take one maintenance algorithm to dominate another if and only if it both deletes more and is more accurate. This finds a set of algorithms which are not bettered by other algorithms and which are all either equal to one another or incomparable with one another (e.g. because one of them deletes more, but the other has higher accuracy).

In Figure 4, we plot all 307 of our algorithms. The percentage of cases deleted is on the x-axis, and the percentage accuracy is on the y-axis. Each point represents one algorithm: red squares are algorithms on the Pareto front; blue diamonds are algorithms that are dominated by those in the Pareto front.

The Pareto front in Figure 4 contains algorithms that delete everything and hence have extremely low accuracy, and vice versa. However, it also contains algorithms with a good balance between the two. These are very easy to identify on the graph because we can see where accuracy begins to fall rapidly. This is the point where 80% or more of the case-base is deleted. We have circled these algorithms on the graph. The four algorithms we have circled are: {{RENN, BBNR}^{U}, RCR}^{S}, BBNR→RCR, RENN→RCR and {RENN, BBNR, RCR}^{S}. Again, we can see that sequences and committees containing RCR and the noise algorithms perform strongly.

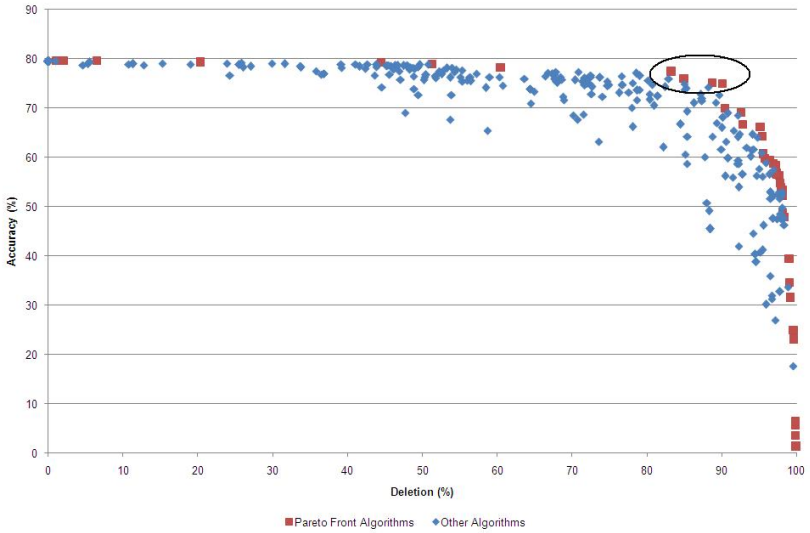

Fig. 4. Result on 27 different datasets, highlighting the Pareto front

The advantage of the graph is that a case-base administrator can investigate the compromises that need to be made. For example, if she wants to delete 50% of the case-base, the graph shows that this can be done while retaining 78% accuracy; but if she wants to delete 90%, then accuracy drops to 75%. Similarly, if she wants to keep accuracy above 79%, the most that she can delete is about 45%. Of course, to be truly useful, the administrator needs a graph that shows the Pareto front for her particular case-base, not an average over 27 datasets.

6 Noise-Filtering

The idea of using an initial noise reduction phase followed by a redundancy reduction phase probably originates with Wilson et al [16]. All three classic case-base maintenance algorithms follow suit. Here, inspired by the numerous alternatives that the MACE approach defines, we wanted to determine how beneficial a noise-filtering phase is. Accordingly, in Table 5, we compare all five atomic algorithms, all three classic composite algorithms, algorithms in which atomic redundancy reduction algorithms are paired with noise reduction algorithms that they have never previously been paired with (e.g. BBNR→RCR), and algorithms in which the noise reduction phase comes after the redundancy reduction phase, rather than before it (e.g. RCR→RENN). In addition to averages over all 27 datasets, we look separately at the Breathalyser dataset, which is known to be very noisy (since it was collected in Dublin pubs!), and the Lenses dataset, which is known to be noise-free. As Table 3 shows, the accuracy of these datasets is 79.46%, 71.60% and 72.50%, respectively without maintenance.

	27 datasets		Breathalyser		Lenses	
Algorithm	Del (%)	Acc (%)	Del (%)	Acc (%)	Del (%)	Acc (%)
RENN	24.27	76.57	25.84	68.80	36.25	52.50
BBNR	23.85	**79.06**	25.97	71.60	41.25	60.00
ICFR	60.77	74.60	61.30	66.00	40.00	67.50
RCR	76.67	76.40	77.40	70.80	63.75	**72.50**
CRR	35.76	77.51	41.17	72.00	32.50	**72.50**
RENN→ICFR	78.76	73.58	77.53	**74.00**	44.38	52.50
RENN→RCR	88.75	75.09	87.66	66.80	86.25	55.00
RENN→CRR	60.27	76.23	66.62	67.20	71.25	55.00
BBNR→ICFR	74.95	74.69	72.08	67.20	70.00	55.00
BBNR→RCR	84.99	75.90	85.19	66.80	80.00	65.00
BBNR→CRR	55.31	77.67	61.95	71.20	68.13	65.00
ICFR→RENN	85.43	64.20	87.79	50.80	79.38	45.00
RCR→RENN	**91.65**	65.46	**92.60**	52.00	**92.50**	45.00
CRR→RENN	64.95	73.35	72.73	62.40	75.63	47.50
ICFR→BBNR	81.47	72.39	80.52	59.20	76.88	57.50
RCR→BBNR	89.67	72.63	87.79	69.20	91.88	45.00
CRR→BBNR	60.46	78.20	64.68	70.00	75.63	62.50

Table 5. Results for different uses of noise reduction algorithms

Firstly we look at the atomic algorithms and their performance over the datasets. We see that the atomic *noise* reduction algorithms (RENN, BBNR) never actually improve accuracy, even on the noisy Breathalyser data (although BBNR does maintain the same accuracy here while deleting 25.97%). We also see that they both delete a large proportion of the Lenses dataset even though this is noise-free, and they both cause large accuracy drops as a result. Unexpectedly, all three atomic *redundancy* reduction algorithms have higher accuracy than the noise reduction algorithms on this dataset. In fact, CRR has a higher accuracy than RENN across the table, and only loses to BBNR on the results averaged over the 27 datasets. RCR has similar accuracy to both RENN and BBNR on the Breathalyser dataset while deleting over 50% more of the case-base.

Secondly, we look at the changes in accuracy and deletion when each of the noise reduction algorithms is run before each of the redundancy reduction algorithms. For each such algorithm, on the Lenses dataset, there is a large drop in accuracy. This is not surprising given the fact that the atomic noise reduction algorithms performed so badly on this dataset. For the Breathalyser dataset, only RENN→ICFR increases accuracy; BBNR→CRR has the smallest fall in accuracy and yet deletes the most. In the results averaged over the 27 datasets, changes in accuracy are quite small. Using BBNR in the noise reduction phase gives slightly less loss of accuracy than using RENN.

When we look at the deletion results, we can see that these composites do delete more than their constituents on their own, as we would expect. Mostly, deleting more cases lowers accuracy. But there are exceptions where accuracy

improves (e.g. RENN→ICFR on the Breathalyser dataset) and where accuracy falls only a little while the case-base shrinks a lot (e.g. BBNR→CRR on the Breathalyser dataset). Interestingly, these good results come from 'classic' algorithms (RENN→ICFR, BBNR→CRR). But there are novel combinations that are doing well on the average results, e.g. RENN→CRR, which might be worthy of investigation on other individual datasets.

Finally, we look at the effect of switching around the sequences so that the redundancy reduction algorithm comes before the noise reduction algorithm. The results here are very consistent. In all situations, the resulting algorithm deletes a greater amount than it does in the conventional ordering. On the other hand, the resulting algorithm is always less accurate than its original with three exceptions: CRR→BBNR is more accurate than BBNR→CRR on the averaged data, RCR→BBNR is more accurate than BBNR→RCR on the Breathalyser data, and ICFR→BBNR is more accurate than BBNR→ICFR on the Lenses dataset. Of these, RCR→BBNR might be worthy of further investigation.

In summary, this analysis shows that the noise reduction phase that is used by so many case-base maintenance algorithms is not always useful, and in some cases may be quite detrimental to the accuracy of the algorithm. It also shows that the three classic composite algorithms do not necessarily use the best algorithm for their noise reduction phase, and that the best algorithm to use can change depending on the dataset.

7 The Effect of Boundary Complexity

The complexity of the boundary between classes in a dataset may have an effect on the performance of the different maintenance algorithms. For example, as noted previously, since RCR retains a case c if it is surrounded by cases of the same class as c, rather than retaining boundary cases, it may cause a loss of accuracy when boundaries are complex. To explore this idea, we computed a boundary complexity measure on the datasets. The boundary complexity measure that we use is the intra/inter class distance ratio [8]. We looked at the two datasets with highest complexity (Lettings and Flags) and the two with lowest complexity (Zoo and Iris) according to this measure. For each of these datasets, we found the top five algorithms ordered by harmonic mean. These top five algorithms are shown in Tables 6 and 7.

We can see that the top five algorithms for the datasets with highest complexity are quite similar. Almost all of them are committees of two sequences, where the sequences comprise a noise reduction algorithm along with either RCR or CRR. The top algorithms for the datasets with lowest complexity are also very similar to each other, with three algorithms common to both datasets. We can see that these algorithms are much more like the ones that did well overall (Section 5) containing one or both of the noise reduction algorithms along with RCR. We also note that the top algorithms for the complex datasets are quite different to the top algorithms for the simple datasets. Interestingly too, the classic RENN→RCR algorithm is in the top three for both the Zoo and the

Lettings	Flags
{BBNR→CRR, RENN→RCR}^{S}	{RENN→CRR, CRR→BBNR}^{S}
{RCR→BBNR, CRR→RENN}^{S}	{RENN, BBNR, CRR}^{S}
{CRR→RENN, RCR→BBNR}^{S}	{RENN→RCR, RCR→BBNR}^{U}
{RENN→RCR, CRR→BBNR}^{S}	{RENN→CRR, RCR→RENN}^{U}
{RCR→RENN, RENN→ICFR}^{S}	{CRR→RENN, RCR→BBNR}^{U}

Table 6. Top five algorithms ordered by harmonic mean for most complex datasets

Zoo	Iris
RENN→RCR	{RENN, RCR}^{S}
{RENN, BBNR, RCR}^{S}	{RENN, BBNR, RCR}^{S}
{RENN→RCR, RCR→RENN}^{U}	RENN→RCR
{{RENN, BBNR}^{U}, RCR}^{S}	{{RENN, BBNR}^{U}, RCR}^{S}
RCR	BBNR→RCR

Table 7. Top five algorithms ordered by harmonic mean for least complex datasets

Iris datasets. However, it comes in 16th place for the Flags dataset, and in 52nd place for the Lettings dataset. This suggests that there is a need for investigation of maintenance algorithms that are suited to datasets with complex boundaries.

8 The Special Case of Spam

Spam-filtering is a task with special characteristics [5]. Of particular relevance here are the facts that spam is heterogeneous (hence, it is a disjunctive concept), and there is a high cost of false positives. We decided, therefore, to look separately at how the maintenance algorithms perform on our five spam datasets.

As well as recording the percentages of accuracy and deletion for each of the algorithms, we also recorded the rate of false positives, the rate of false negatives and the within-class error rate (the average of the other two rates) [5]. Table 8 shows the best five algorithms, ordered by increasing within-class error rate.

We can see that the algorithms that do well on the spam datasets are quite different from the algorithms that have done well on other datasets. BBNR performs very strongly, with the lowest rate of error overall. It also provides the noise removal component of all of the sequences and committees in the top five; RENN is not contained in any of the top five. Since BBNR was developed specifically to remove noise from spam datasets, this result is not surprising. However, it does confirm the strength of the algorithm for this domain.

It is also interesting to note that CRR, the algorithm developed specifically to remove redundancy from spam datasets, does not perform as strongly. It is contained in three of the top five algorithms, but appears less often than RCR. Also, the 'classic' BBNR→CRR composite comes only in 62nd place.

Algorithm	Del	Acc	FP Rate	FN Rate	Err Rate
BBNR	5.84	96.16	2.64	5.02	3.82
$\{\{$CRR, RCR, ICFR$\}^U$, BBNR$\}^S$	35.58	95.86	2.42	5.84	4.14
$\{$RCR, BBNR$\}^U$	4.54	95.64	2.46	6.26	4.36
$\{$BBNR→CRR, BBNR→ICFR, ICFR→BBNR, CRR→BBNR, BBNR→RCR, RCR→BBNR$\}^U$	28.88	95.60	2.44	6.30	4.38
$\{$CRR→BBNR, BBNR→RCR$\}^U$	38.14	95.56	2.22	6.66	4.44

Table 8. Top five algorithms ordered by within-class error rate for the spam datasets

This indicates that, while the BBNR part of the composite algorithm is well suited to spam datasets, CRR is not as well suited. In fact, it appears that no single redundancy removal algorithm on its own deals well with the spam datasets. The committees in the top five all contain at least two atomic redundancy removal algorithms, if not all three. This may be due to the fact that spam is heterogeneous; it is more difficult to be sure that spam cases are redundant because they are distributed quite widely across the case base.

9 Conclusions and Future Work

In this paper we have investigated the most commonly-used case-base maintenance algorithms and have shown their strengths and weaknesses. We presented our MACE approach, which allows us to combine these algorithms. We investigated the performance of 307 algorithms defined by MACE using 27 datasets.

Our MACE algorithms performed strongly: four of the top five algorithms were new sequences or committees. As well as reporting the top algorithms over the 27 datasets, we looked at three particular areas where results could be different. We examined the initial noise reduction phase that the classic algorithms use and concluded that it is not always beneficial. We looked at boundary complexity and the effect that this has on the maintenance algorithms. We showed that the RCR algorithm, which works well on simple datasets, performs less well on those with complex boundaries. We also looked at the spam domain and showed that the BBNR algorithm does very well on this domain.

Our ongoing work consists of predicting a good maintenance algorithm for a given dataset based on properties of that dataset. In particular, we are investigating using a 'meta-case-base' for this task.

References

1. A. Asuncion and D.J. Newman. UCI Machine Learning Repository, 2007.
2. H. Brighton and C. Mellish. On the Consistency of Information Filters for Lazy Learning Algorithms. In J. Rauch and J.M. Zytkow, editors, *Procs. of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, pages 283–288. Springer-Verlag, 1999.

3. C.E. Brodley and M.A. Friedl. Identifying and Eliminating Mislabeled Training Instances. In *In AAAI/IAAI*, pages 799–805. AAAI Press, 1996.
4. P. Cunningham and G. Zenobi. Case Representation Issues for Case-Based Reasoning from Ensemble Research. In D.W. Aha and I. Watson, editors, *Procs. of the 4th International Conference on Case-Based Reasoning*, LNCS-2080, pages 146–157. Springer-Verlag, 2001.
5. S.J. Delany and P. Cunningham. An Analysis of Case-Based Editing in a Spam Filtering System. In P. Funk and P. González-Calero, editors, *Procs. of the 7th European Conference on Case-Based Reasoning*, LNCS-3155, pages 128–141. Springer, 2004.
6. T.G. Dietterich. Ensemble Methods in Machine Learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, LBCS-1857, pages 1–15. Springer, 2000.
7. D. Doyle, P. Cunningham, D.G. Bridge, and Y. Rahman. Explanation Oriented Retrieval. In P. Funk and P. González-Calero, editors, *Procs. of the 7th European Conference on Case-Based Reasoning*, LNCS-3155, pages 157–168. Springer, 2004.
8. T. K. Ho and M. Basu. Measuring the Complexity of Classification Problems. In *Procs. of the 15th International Conference on Pattern Recognition, 2000*, pages 43–47, 2000.
9. D.B. Leake and R. Sooriamurthi. When Two Case Bases Are Better than One: Exploiting Multiple Case Bases. In D.W. Aha and I. Watson, editors, *Procs. of the 4th International Conference on Case-Based Reasoning*, LNCS-2080, pages 321–335. Springer-Verlag, 2001.
10. E. McKenna and B. Smyth. Competence-Guided Case-Base Editing Techniques. In E. Blanzieri and L. Portinale, editors, *Procs. of the 5th European Workshop on Case-Based Reasoning*, LCS-1898, pages 186–197. Springer-Verlag, 2000.
11. R. Nicholson, D. Bridge, and N. Wilson. Decision Diagrams: Fast and Flexible Support for Case Retrieval and Recommendation. In T. Roth-Berghofer, M.H. Göker, and H. Altay Güvenir, editors, *Procs. of the 8th European Conference on Case-Based Reasoning*, LNAI-4106, pages 136–150. Springer, 2006.
12. A. Orecchioni, N. Wiratunga, S. Massie, and S. Craw. k-NN Aggregation with a Stacked Email Representation. In K. D. Althoff, R. Bergmann, M. Minor, and A. Hanft, editors, *Procs. of the 9th European Conference on Case-Based Reasoning*, LNCSS-5239, pages 415–429. Springer-Verlag, 2008.
13. E. Plaza and L. McGinty. Distributed Case-Based Reasoning. *The Knowledge Engineering Review*, 20(3):261–265, 2006.
14. R.J. Quinlan. Bagging, Boosting, and C4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730. AAAI Press, 1996.
15. I. Tomek. An Experiment with the Edited Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):448–452, 1976.
16. D.R. Wilson and T.R. Martinez. Instance Pruning Techniques. In D. Fisher, editor, *Procs. of the 14th International Conference on Machine Learning*, pages 403–411. ICML, Morgan Kaufmann, 1997.
17. N. Wiratunga, S. Craw, and Ray. Rowe. Learning to Adapt for Case-Based Design. In S. Craw and A. D. Preece, editors, *Procs. of the 6th European Conference on Case-Based Reasoning*, LNCS-2146, pages 421–435. Springer-Verlag, 2002.