Complexity-Guided Case Discovery for Case Based Reasoning

Stewart Massie and **Susan Craw** and **Nirmalie Wiratunga**
The Robert Gordon University
Aberdeen, AB25 1HG, Scotland, UK
sm@comp.rgu.ac.uk smc@comp.rgu.ac.uk nw@comp.rgu.ac.uk

Abstract

The distribution of cases in the case base is critical to the performance of a Case Based Reasoning system. The case author is given little support in the positioning of new cases during the development stage of a case base. In this paper we argue that classification boundaries represent important regions of the problem space. They are used to identify locations where new cases should be acquired. We introduce two complexity-guided algorithms which use a local complexity measure and boundary identification techniques to actively discover cases close to boundaries. The ability of these algorithms to discover new cases that significantly improve the accuracy of case bases is demonstrated on five public domain classification datasets.

Introduction

Case Based Reasoning (CBR) solves new problems by reusing the solution of previously solved problems. The case base is the main source of knowledge in a CBR system and, hence, the availability of cases is crucial to a system's performance. It is the availability of cases that often supports the choice of CBR for problem-solving tasks, however, in real environments there are often gaps in the coverage of the case base because it is difficult to obtain a collection of cases to cover all problem-solving situations.

Adaptation knowledge can be used to provide solutions to new problems that occur in the gaps that result from a lack of case coverage. However, gaining effective adaptation knowledge may be impossible or require considerable knowledge acquisition effort. The inclusion of additional, strategically placed cases can provide a more cost-effective solution.

Case discovery is the process of identifying *useful* new cases to fill gaps that exist in the coverage of the case base. This is different from traditional case learning, through the retain stage of the CBR cycle, in which newly solved problems are routinely added to the case base to assist future problem-solving. Rather case discovery is an active learning problem in which the aim is to identify areas of the problem space in which new cases would help to improve the system's performance and to create cases to fill these gaps. Commercial systems generally assume that a suitable case base already exists and give little help to the case author in the case discovery stage of building the case base. There is a need for techniques to assist the case author during this crucial case base development stage.

We argue that new cases should be placed in regions of the problem-space in which the system is uncertain of the solution, and that these regions are generally close to boundaries between classifications. In this paper we present a new technique to identify and rank these areas of uncertainty and create candidate cases to assist the case author place new cases in these regions.

The remainder of this paper describes our approach and evaluates it on several public domain case bases. The next section discusses existing work on case discovery. The following sections outline how we use a complexity metric, boundary detection and clustering to identify areas of the problem-space that need the support of new cases and how these cases are created. The approach is then evaluated against two benchmark algorithms before we draw some final conclusions.

Related Work in Case Discovery

The case discovery problem can be considered in two stages. First *interesting* areas or gaps within the coverage of the case base must be identified and secondly cases must be created to fill these gaps. This presents a more complex challenge when compared to the more commonly researched case base editing or selective sampling problems that have a pool of existing cases from which to select cases. In contrast, the task of case discovery is to add to the case knowledge using implicit information held within the case base.

Some research has focused on the first stage of the discovery process. One approach to identifying gaps has been to focus on locating maximal empty hyper-rectangles within k-dimensional space (Liu, Ku, & Hsu 1997). In their later research the algorithm is capable of locating hyper-rectangles within data containing both continuous and discrete valued attributes (Liu *et al.* 1998). The main problem with this approach is that there is no way to identify if the gap found in the problem space is *interesting* from a problem-solving view-point, or even represents a possible combination of attribute values. An alternative approach to identifying interesting areas for new cases is proposed in (Wiratunga, Craw, & Massie 2003) as part of a selective sampling technique.

In this approach the case base plus unlabelled examples are formed into clusters using a decision tree technique. The clusters are then ranked based on the mixture of classes present, and then unlabelled examples are ranked based on their distance from labelled cases and closeness to other unlabelled examples. However this approach draws on knowledge gained from a pool of unlabelled cases not normally available during the case discovery process.

CaseMaker (McSherry 2001) is a knowledge acquisition tool that addresses both stages of the discovery process. A complete set of all uncovered cases is found by exhaustively searching the space of allowable feature combinations. These cases are ranked based on their potential coverage contributions, calculated using an adaptation function based on feature differences. This approach has been shown to be successful in finite domains where suitable domain knowledge and adaptation knowledge are available.

Competence-guided case discovery is an alternative approach based on a competence model (McKenna & Smyth 2001). The model groups together cases that solve each other into clusters called competence groups (Smyth & McKenna 2001). For each pair of competence groups the two closest cases are identified as the boundary pair and gaps are identified as the space between these *nearest neighbour* competence groups. Starting with the smallest gap a case is created whose feature values are determined by the cases in the neighbourhood of the boundary pair. While this intuitive approach is likely to discover valid cases in active regions of the problem space it gives no guarantee on finding the *most interesting* gap as it ignores large parts of the problem space.

Exploiting the existing knowledge within the case base is common to all the approaches discussed here and, likewise, we use this knowledge source to identify areas of uncertainty within the problem space and then to identify cases on classification boundaries.

Complexity-Guided Case Discovery

Our aim is to discover cases that improve the CBR system's accuracy. We believe cases close to classification boundaries are most likely to achieve this aim. As discussed earlier, the case discovery problem can be considered in two stages: identification of interesting areas of the problem space in which to place new cases is discussed in this section while the creation of new cases to fill these gaps is discussed in the following section.

Previous research on case base editing has highlighted the importance of cases in boundary regions for the competence of a case base (Brighton & Mellish 2002; Wilson & Martinez 2000). It seems reasonable to expect a successful case creation algorithm to also identify cases on class boundaries. Our approach to identifying where new cases should be placed, in order to improve a system's accuracy, involves several stages that combine to identify boundary cases.

Areas of Uncertainty The first stage in finding interesting areas for new cases is to find areas in which cases are likely to be wrongly classified. We do this by using a local complexity metric.

Classification complexity is an inherent problem characteristic that gives a measure of how difficult it is to classify new problems. It is determined by such factors as the overlap and length of decision boundaries, the dimensionality of the feature space, the noise level and the sparseness of available cases. Accuracy gives one measure of complexity but is dependent on the classifier chosen and provides no local information on areas of complexity within the problem space.

Several approaches have been used to estimate overall classification complexity. However, in case discovery we are interested in the complexity at local areas. We have chosen an approach that allows us to measure the local complexity based on the spatial distribution of cases rather than on a probabilistic distribution. In this approach the complexity of each case is calculated using a metric based on its k-Nearest Neighbours while incrementally increasing the value of k.

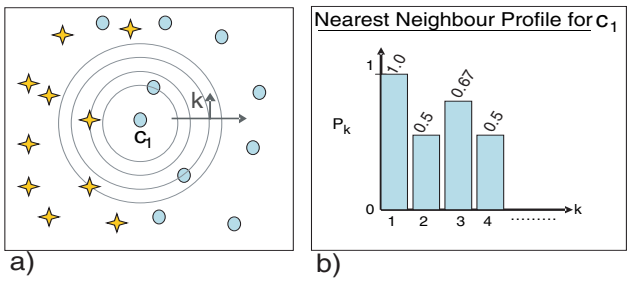

Figure 1: Complexity metric calculation

The complexity measure is calculated for each case by looking at the class distribution of cases within its local neighbourhood. P_k is the proportion of cases within a case's k nearest neighbours that belong to the same class as itself. In Figure 1a, as the value of k increases, the sequence of P_k starts 1, 0.5, 0.67, 0.5. A nearest neighbour profile can now be plotted of P_k as k increases. The complexity metric used is the area of the graph under the profile with the x-axis normalised, shown by the shaded area on Figure 1b. Case complexity is calculated by

$$\text{complexity} = 1 - \frac{1}{K}\sum_{k=1}^{K} P_k$$

for some chosen K. With K=4 the complexity of c_1 is 0.33. As the metric is weighted to a case's nearest neighbours using a large value for K has little impact on the results and K=10 was used in our calculations.

Cases with high complexity are close to classification boundaries and identify areas of uncertainty within the problem space. The regions around these target cases are identified as requiring support. Target cases are ranked in descending order of complexity to prioritise between the different regions.

Class Boundaries The case complexity metric is used to identify target cases in regions of the problem space near classification boundaries that we believe would benefit from the support of additional cases. However, it gives no help on

where, within these regions, the new cases should be placed. Following our hypothesis that cases close to class boundaries are important in case discovery we want to discover cases closer to the boundaries. There must be a classification boundary in the problem space close to the target case, however its direction and location are not known. To find an outer limit for the location of the boundary the target case's nearest unlike neighbour (NUN) is found i.e. the nearest case that has a different class. The boundary lies between these two reference cases i.e. the target case and its NUN.

Clustering Prioritising regions of the problem space using only the complexity value of cases is expected to identify interesting areas in which additional cases will improve the system's accuracy. However, prioritising on case complexity alone potentially gives two problems. There is a danger that new cases will be concentrated in a small region of the problem space as high complexity cases are likely to be located close to each other. In addition, new cases may be concentrated on small pockets of cases whose classification is different to their neighbours, as these cases will have high complexity values, resulting in poorer performance in noisy or multi-class problems.

Partitioning the case base into clusters may give a more balanced distribution of discovered cases over the whole case base. Competence group clustering (Smyth & Keane 1995) is a commonly used clustering technique in CBR and a similar approach has been adopted here. Clusters are formed using leave-one-out testing to measure the problem-solving ability of a case using: coverage and reachability. Coverage of a case is the set of problems that case can solve; conversely, reachability is the set of all cases that can solve it. Next clusters of cases, called competence groups, are formed using their reachability and coverage sets to group cases that have overlapping sets. This clustering model is typically applied to CBR systems incorporating an adaptation stage, however, here it is being applied to retrieve-only classification. In this scenario, the reachability set of a case is its k-nearest neighbours with the same classification but bound to the first case of a different class (Brighton & Mellish 1999).

With the case base formed into clusters, the complexity of each cluster can be defined as the average complexity of the cases it contains. The clusters can be ranked in descending order of complexity. Now, rather than choosing target cases purely on complexity ranking, one case can be chosen from each cluster with cluster complexity used to prioritise the target cases. The target case chosen from each cluster is the case with the highest complexity. In addition, there is now the opportunity to remodel the case base, by reforming the clusters as new cases are added, and building the effect of the discovered cases into the next round of case discovery.

Creating a New Case

The methods described in the previous section are used to identify interesting areas of the problem space for new cases. The second stage of the case-discovery process is to create a *candidate* case to occupy the area between the two reference cases. This involves setting suitable feature values for the candidate case.

Candidate Case Feature Values Two approaches for setting the candidate case's feature values were investigated. In the first, the feature values are set as either the mean (numeric features) or majority (nominal features) of the feature values of the reference cases and their related sets. A case's related set is the union of its coverage and reachability sets. This approach, used by McKenna & Smyth, was found not to work well in domains containing small groups of exceptional cases. This may be due to one of the reference cases coming from a much larger competence group and applying excessive influence on the feature values, and hence location, of the candidate case. An alternative simpler approach was found to give more consistent results and was adopted for the complexity-guided algorithms. In this simpler approach the candidate case uses only the boundary pair to set its problem feature values. This results in a discovered case more evenly spaced between the reference pair.

Case discovery aims to create a new case for inclusion in the case base. Inclusion of the candidate case may be automatic but, as there is no guarantee that a candidate case will be a valid case occupying an active region of the problem space, the more likely scenario is for the case author to validate the case prior to its inclusion in the case base.

Noise Filter A potential problem of discovering cases on classification boundaries is that noisy cases may be discovered in domains containing significant levels of noise or exceptional cases. Indeed, most modern case editing algorithms (Brighton & Mellish 2002; Delany & Cunningham 2004) apply noise reduction algorithms prior to an editing approach that retains boundary cases.

A typical approach to noise reduction is to remove cases that are incorrectly classified (Wilson 1972). We apply a similar approach to determine if a validated case should be included in the case base. A *friend* to *enemy* distance ratio is calculated using the similarity metric. The enemy distance is the average distance within the case base to the validated case's three NUN's whereas the friend distance is the average distance to the validated case's three nearest like neighbours. A high ratio indicates a validated case that may harm the system's accuracy and would not be included in the case base. A conservative or aggressive approach to noise filtering can be applied by varying the ratio above which a validated case is not added to the case base. Noise filtering has only been used on known noisy datasets and has been applied using a conservative approach by not accepting validated cases with a ratio greater than 1.5.

Evaluation

In order to confirm that complexity-guided case discovery is useful we need to demonstrate that *useful* cases are discovered. Two complexity-guided algorithms have been compared with two benchmark algorithms to determine whether they result in case bases with increased accuracy. Five public domain classification datasets from the UCI ML repository (Blake, Keogh, & Merz 1998) have been used in the

evaluation. The selected datasets have varying numbers of features and classes, proportion of nominal to numeric attributes and level of noise. In addition, some datasets have missing values.

Complexity-guided case discovery cannot guarantee valid cases will be discovered. The objective is to supply a complete, candidate case to the case author to either accept or create a slight variation that corresponds to a valid case. However, this situation is difficult to replicate in an experimental evaluation because a domain expert is not available to validate the discovered cases. To simulate an expert our experimental design uses a pool of independent cases to act as an oracle. The candidate case then acts as a probe into this pool of cases to retrieve the most similar case from the oracle.

Each dataset was split into 5 independent folds with the folds being stratified to ensure a proportional representation of each class in each fold. One fold was used as the training set, one of the remaining four folds was used as the the test set with the pool of cases being made up of the three unallocated folds. This process was repeated for each combination in turn resulting in 20 experiments (unique combinations of training set, test set and pool cases) for each dataset. There was no overlap between a training set and its associated test set and pool of cases.

The case base was initialised by randomly selecting a fixed number of cases from the training set. The starting size of the case base varied between 10 and 35 cases, depending on the dataset size and the difficulty of the problem. The algorithms were run on each trial on each dataset to discover between 5 and 40 cases in steps of 5. The results, averaged over the 20 runs, are plotted as a graph of the average accuracy for the test set for an increasing case base size, as an increasing number of cases are discovered. Test set accuracy is evaluated by a straightforward *k*-NN.

The experiments evaluate the effectiveness of the four algorithms described below on test set accuracy with a varying number of cases being discovered.

Algorithms

Four different case-discovery techniques have been implemented. Two are complexity-guided algorithms using a combination of the previously discussed techniques while the remaining algorithms provide benchmark comparisons.

All the algorithms identify two reference cases (a target case and its pair case) from within the case base. The main difference between the four algorithms is in their approach to identifying these reference cases.

- **COMPLEXITY** is our simpler complexity-guided algorithm. The complexity metric is calculated for each case and the 50% of cases with the highest complexity are ranked in descending order. Each case in turn (until the desired number of cases are discovered) is selected as the target case and its NUN is identified as its pair case. These two reference cases are used to create a candidate case to lie between them by setting the candidate's feature values as either the mean or majority of the reference cases' feature values.
- **COMPLEXITY+** is a more informed algorithm that uses clustering to create a model of the case base. Figure 2 shows a simplified view of how this algorithm works in 2 dimensions. There are cases belonging to two classes with a class boundary between them. The cases are formed into clusters and the case with the highest complexity in each cluster is chosen as the target case (shown as a solid case). The target case's NUN is found (shown by an arrow) giving two reference cases and a candidate case is created to lie between them, as shown by the square.

  The implementation of the algorithm involves the following stages. The complexity metric is calculated for each case. Clusters are formed and their complexity calculated as discussed earlier. The 75% of clusters with highest complexity are ranked in descending order of complexity. A target case is selected from each cluster in turn (until the required number of cases are discovered) and its NUN is selected as its pair case. A candidate case is created in the same manner as in COMPLEXITY. Where more cases are required than available clusters the stages are repeated including the complexity calculation and clustering. This incorporates the effect of the already discovered cases into the model.
- **COMPETENCE** uses competence-guided case discovery to create new cases between the nearest neighbour competence groups (McKenna & Smyth 2001). Two reference cases are selected from different competence groups that are nearest to each other. The candidate case's feature values are set using the feature values of the reference cases' related sets.
- **RANDOM** is an uninformed algorithm that selects two reference cases at random from the case base and then uses these reference cases to create a candidate case in the same way as COMPLEXITY. This process continues until the required number of cases have been discovered.

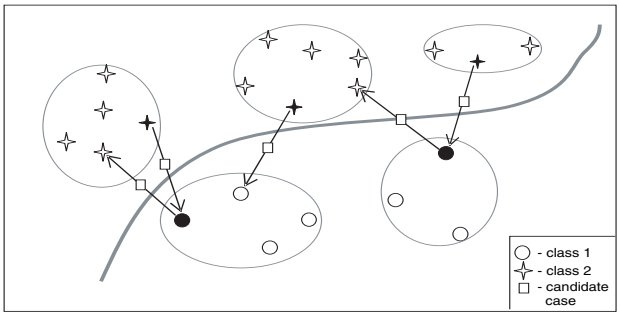

Figure 2: Illustration of COMPLEXITY+

Results

Significance is reported from a one-tailed paired t-test at 99% confidence, unless otherwise specified. Figure 3 (a) and (b) show average accuracy results for each case base size on the House Votes and Hepatitis domains. These are both binary classification problems with a bias to one of the classes. House votes is the larger data set with 435 cases

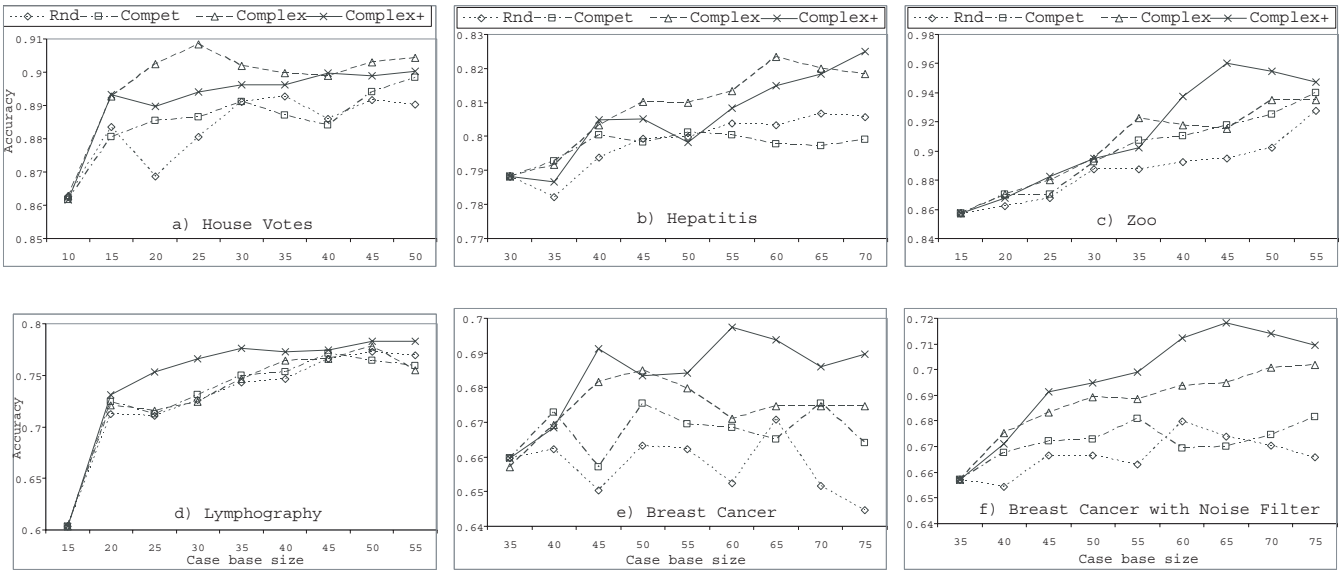

Figure 3: Accuracy of Growing Case Bases as Cases are Discovered

containing 17 nominal features while Hepatitis is a smaller data set of 155 cases represented by 20 mostly nominal features containing some missing values. As expected we see a significant improvement in accuracies on both House Votes and Hepatitis by the two complexity-guided algorithms (COMPLEXITY and COMPLEXITY+) over the RANDOM and COMPETENCE. Perhaps surprisingly, the simpler COMPLEXITY gives the best performance on these datasets. This might be explained by these being binary problems with high accuracy suggesting a more simple boundary than on the other datasets. The simpler algorithm, by concentrating on only few areas of the problem space, appears to perform well on this type of domain.

Average accuracy for the Zoo and Lymphography domains appear in Figure 3 (c) and (d). These are multi-class problems: Zoo has 101 cases split between 7 classes while Lymphography has 148 cases covering 4 classes. These domains have similar number of features (18 and 19) with no missing values. Zoo contains only nominal features whereas Lymphography contains both nominal and numeric. In both these domains COMPLEXITY+ produces the best performance with significant improvement over the other three algorithms. COMPLEXITY shows a significant improvement over RANDOM on the zoo domain but no difference over COMPETENCE. On Lymphography COMPLEXITY gave no improvement over either benchmark algorithm. The relatively poor performance of COMPLEXITY might be expected on these multi-class domains, as some of the classes contain a very small number of cases. In these situations COMPLEXITY will concentrate on providing cases to support the classes with low representation and provide insufficient support to the rest of the problem space. In contrast, COMPLEXITY+ uses clustering to provide a more balanced distribution of new cases.

Figure 3 (e) shows average accuracy results on the Breast Cancer dataset. In Figure 3 (f) a noise filter, as described earlier, has been applied to all four algorithms for Breast Cancer. This is a binary classed domain with 9 multi-valued features containing missing data. The noise filter has been added because Breast Cancer is a more complex domain containing either noise or exceptional cases resulting in lower accuracies than the other domains. COMPLEXITY+ again produces the best performance with significant improvements over the other three algorithms. COMPLEXITY also shows a significant improvement over the two comparison algorithms in both experiments although the improvement over COMPETENCE without the noise filter is only significant at 95% confidence. The improved performance of COMPLEXITY+ over COMPLEXITY might again be explained by the simpler algorithm concentrating on supporting the noise or exceptional cases. It is interesting to see that, although the noise filter results in a small improvement in the performance of the benchmark algorithms it gives a large and significant improvement to the accuracies achieved by both the complexity-guided algorithms. This improvement is to be expected in noisy datasets because, by choosing cases on class boundaries, the complexity-guided algorithms will have a greater tendency to pick noisy cases.

Evaluation Summary

The results from the significance tests, comparing the two complexity-guided case discovery algorithms with the benchmark algorithms on each dataset, are summarised in Table . The first two columns display the improvement with COMPLEXITY while the other two columns show significance results for COMPLEXITY+.

Overall COMPLEXITY+'s performance shows a significant improvement over the comparison algorithms on all the

Data Set	COMPLEXITY vs. RANDOM	COMPLEXITY vs. COMPETENCE	COMPLEXITY+ vs. RANDOM	COMPLEXITY+ vs. COMPETENCE
House Votes	√	√	√	√
Hepatitis	√	√	√	√
Zoo	√	no difference	√	√
Lymphography	no difference	no difference	√	√
Breast Cancer	√	√ (95%)	√	√
Breast Cancer-Noise	√	√	√	√

Table 1: Results summary according to significance

datasets and it provides the most consistent approach to case discovery of the algorithms studied. COMPLEXITY is shown to perform well on binary problems, particularly on simpler problems and on domains with low levels of noise, however, its performance on multi-class problems is only comparable with the benchmark algorithms.

The introduction of a noise filter stage gave significant accuracy improvements on the two complexity-guided discovery algorithms with Breast Cancer. This highlights the importance of noise filtering in noisy datasets.

Conclusions

The novel contribution of this paper is the use of a complexity metric and a case's NUN to guide the case discovery process by identifying interesting areas of the problem space. The idea of placing new cases on classification boundaries appears to be intuitively sensible in that it mirrors the approach of recently developed case base editing algorithms. COMPLEXITY and COMPLEXITY+, two new complexity-guided algorithms, were introduced and their effectiveness was demonstrated on 5 public domain datasets. In general, a significant improvement in test accuracy was observed with these new techniques compared to the random and competence-guided algorithms used as benchmarks. COMPLEXITY performed well on simple binary domains but suffered on multi class problems or on datasets containing noise. COMPLEXITY+, which incorporated a clustering stage, provided the most consistent performance across the range of datasets. A noise filter stage was found to enhance the performance of COMPLEXITY and COMPLEXITY+ on noisy datasets.

One limitation of the complexity-guided algorithms is that they restrict their search space to finding new cases within the problem space already covered by existing cases. Future work will focus on developing a complimentary approach for the very early growth stages of a case base, perhaps by using domain knowledge to seed the case base.

In this paper we have concentrated on providing support for the case author in the case discovery problem. However we are keen to see how the use of a complexity measure might be used more generally to provide support to the case author in other case base maintenance areas, such as case editing.

References

Blake, C.; Keogh, E.; and Merz, C. 1998. UCI repository of machine learning databases.

Brighton, H., and Mellish, C. 1999. On the consistency of information filters for lazy learning algorithms. *In Principles of Data Mining and Knowledge Discovery: 3rd European Conf.*, 283–288.

Brighton, H., and Mellish, C. 2002. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery* 6(2):153–172.

Delany, S. J., and Cunningham, P. 2004. An analysis of case-base editing in a spam filtering system. In *Proc. of the 7th European Conf. on Case-Based Reasoning*, 128–141.

Liu, B.; Wang, K.; Mun, L.-F.; and Qi, X.-Z. 1998. Using decision tree induction for discovering holes in data. In *Proc. of the 5th Pacific Rim Int. Conf. on Artificial Intelligence*, 182–193.

Liu, B.; Ku, L.-P.; and Hsu, W. 1997. Discovering interesting holes in data. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence*, 930–935.

McKenna, E., and Smyth, B. 2001. Competence-guided case discovery. In *Proc. of the 21st Int. Conf. on Knowledge Based Systems and Applied Artificial Intelligence*, 97–108.

McSherry, D. 2001. Intelligent case-authoring support in casemaker-2. *Computational Intelligence* 17(2):331–345.

Smyth, B., and Keane, M. T. 1995. Remembering to forget. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence*, 377–382.

Smyth, B., and McKenna, E. 2001. Competence models and the maintenance problem. *Computational Intelligence* 17(2):235–249.

Wilson, D. R., and Martinez, T. R. 2000. Reduction techniques for instance-based learning algorithms. *Machine Learning* 38(3):257–286.

Wilson, D. 1972. Asymptotic properties of nearest neighbour rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics* 2(3):408–421.

Wiratunga, N.; Craw, S.; and Massie, S. 2003. Index driven selective sampling for CBR. In *Proc. of the 5th Int. Conf. on Case-Based Reasoning*, 637–651.