# Competence-guided Editing Methods for Lazy Learning

**Elizabeth McKenna and Barry Smyth**[1]

**Abstract.** Lazy learning algorithms retain their raw training examples and defer all example-processing until problem solving time (eg, case-based learning, instance-based learning, and nearest-neighbour methods). A case-based classifier will typically compare a new target query to every case in its case-base (its raw training data) before deriving a target classification. This can make lazy methods prohibitively costly for large training sets. One way to reduce these costs is to *filter* or *edit* the original training set, to produce a reduced edited set by removing redundant or noisy examples. In this paper we describe and evaluate a new family of hybrid editing techniques that combine many of the features found in more traditional approaches with new techniques for estimating the usefulness of training examples. We demonstrate that these new techniques enjoy superior performance when compared to traditional and state-of-the-art methods.

## 1 Introduction

Inductive learning algorithms are often categorised as either *eager* or *lazy*. The former transforms a set of training examples into an optimised reasoning structure at learning time, which is then used at problem solving time; examples include decision tree learners and neural networks. In contrast lazy learning methods retain the raw training examples and reuse them directly at problem solving time; examples include instance-based learning, case-based learning, and nearest neighbour methods. Lazy learners typically suffer from high problem solving costs. For example, a case-based classifier might compare a new target query to every training example before deriving a target classification. If the training set is large, these costs can be prohibitive. One strategy for reducing these costs is to process the training examples to produce a reduced edited set of examples for the learner to use (see Section 2 for existing methods [1, 3, 4, 5, 6, 7, 8, 12, 13, 14, 15]).

In this paper we focus on lazy learning techniques for classification tasks. We introduce a new family of hybrid editing techniques that combine many of the features found in more traditional approaches with new techniques for modeling and evaluating the usefulness of training examples with respect to their future classification competence (Section 3). We demonstrate, in Section 4, that this new family of techniques benefits from superior editing performance, both in terms of the size and accuracy of the edited sets produced.

## 2 Related Work

There are two basic reasons for editing training data - redundancy reduction and noise removal. Redundancy reduction algorithms aim to eliminate training cases that do not contribute to classification competence, for example, cases from the interior of a densely packed class cluster. These algorithms can produce significant training set reductions but tend to preserve noisy examples as apparent exceptions. In contrast, noise reduction techniques aim to eliminate noisy examples from the training set. These algorithms tend to result in less dramatic size reductions (depending on the level of training set noise) and they can also mistakenly remove important exceptional cases, which are difficult to distinguish from true noise.

Hart's condensed nearest neighbour (CNN) approach is an early redundancy reduction technique [7]. It builds an edited set from scratch by adding cases that cannot be successfully solved by the edited set built so far; the algorithm makes multiple passes over the training data until no more additions can be made. The technique generates significant size reductions but the quality of the final edited set depends on the presentation order of cases. Specifically, redundant cases, which are examined early on in the editing process, tend to be added to the edited set. Solutions to this problem have been suggested, including a post-processing stage to remove redundant cases from the edited set [6] and a pre-processing step to order cases prior to editing [13].

The Edited Nearest Neighbour (ENN) method is a perfect counterpoint to CNN [14]. ENN is a noise removal technique. It contracts the training set by deleting noisy cases to produce the final edited set. A noisy case is one that cannot be classified by its k nearest neighbours. Tomek [12] introduces repeated ENN (RENN) which makes multiple passes over the training data, applying the ENN rule until no more cases can be deleted.

Recently, a number of authors have described editing algorithms based on work from the case-based reasoning community that attempts to explicitly model the competence properties of cases. Smyth and Keane [9, 10] introduce two important competence properties: the *reachability set* (1) of a case c is the set of cases that can successfully classify c; the *coverage set* (2) of a case c is the set of cases that c can classify. The size and contents of these sets represent the local competences of a case (and we will use these sets as the basis for our competence model in the next section).

$$ReachabilitySet(c \in C) = \{c' \in C : Solves(c', c)\} \quad (1)$$

$$CoverageSet(c \in C) = \{c' \in C : Solves(c, c')\} \quad (2)$$

Smyth and Keane describe how these sets can be used to define categories of cases with different competence contributions (for example, *pivotal* cases make unique competence contributions, whereas *auxiliary* cases make no competence contributions). They describe a case-base editing algorithm that biases pivotal cases and actively deletes auxiliary cases.

Brighton and Mellish [3, 4] have recently adapted this technique for use in classification problems. Their approach, called ICF (Incremental Case Filtering), contracts the training set to produce a final edited set by removing cases whose reachability set size is larger

[1] Smart Media Institute, University College Dublin, Belfield, Dublin 4, Ireland, email: {Elizabeth.McKenna, Barry.Smyth}@ucd.ie

than their coverage set size; a case c is deleted if c is solved by more cases that it can solve itself. Brighton and Mellish demonstrate that ICF achieves superior editing performance when compared to many traditional algorithms, and matches the performance of a number of state-of-the-art techniques.

We have provided a brief summary of editing algorithms spanning the last 4 decades of research. Of course, for space reasons, many important approaches have had to be left out. In particular, for completeness, we encourage the reader to refer to the following additional research: Chang's work on prototype selection [5]; the Instance-Based Learning work of Aha et al. [1]; and finally, the work of Wilson and Martinez [15].

## 3 A Framework for Case Editing

Conventional editing algorithms operate in one of two ways: either an edited set is built from scratch by adding cases to it from the training set; or an edited set is produced by contracting the original training set. In this section we describe a new family of editing techniques that combine these two strategies. The new algorithms combine four important features: (1) an *evaluation policy* for evaluating cases according to their competence contributions (based on their coverage and reachability sets); (2) an *addition rule* to select cases from the training set for addition to the edited set; (3) a *deletion rule* to remove cases from the training set before the next round of editing; (4) an *update rule* to update the competence model (the coverage and reachability sets) after each editing step. The template for this family of algorithms is shown in figure 1.

```
T:          Original training cases
CM:         Competence model
Eval:       Ordering function (MCOV/RFC/RC)
Add?:       Use CNN Rule? (T/F)
Delete?: Delete covered cases? (T/F)
Update?: Update CM for remaining cases? (T/F)

1.  Edit(T,CM,Eval,Add?,Delete?,Update?)
2.     T  RENN(T)   {that is, Repeated ENN}
3.     E  {}
4.     While T {} Do
5.        C    Next case in T according to Eval
6.        If Add?
7.           If  Solves?(E,C) then E  E {C}
8.        Else E  E {C}
9.        If Delete? then
10.          E  E-CoverageSet{C}
11.       Else E  E-{C}
12.       If Update? then Update(CM)
13.    EndWhile
14. Return(E)
```

**Figure 1.** The basic algorithmic template for the new family of competence-guided editing techniques. Each algorithmic variation will differ according to the *Eval, Add?, Delete, and Update?* parameters.

### 3.1 Evaluation Policy

The order in which cases are considered during editing can have an important bearing on the quality of the final edited set. For example, we saw earlier that CNN suffers from this problem because its selection rule cannot distinguish between useful and redundant cases during the early stages of editing. One solution is to order cases prior to editing, using an evaluation function to measure the expected usefulness of a case - the next case to be considered during editing is the most useful case left in the training set. We propose three competence-guided evaluation policies that measure different competence properties of cases.

**The Reach for Cover Policy (RFC):** The size of the reachability set of a case can be used as an estimate of how difficult this case is to classify, and thus how important this case is likely to be with respect to classification competence or accuracy. For example, a case with a small reachability set cannot be solved by many other cases, and so this case may be a crucial element of the edited set. The *reach for cover* (RFC) evaluation function implements this idea: the usefulness of a case is an inverse function of its reachability set size. Thus, cases with small reachability sets are considered for membership in the edited set before cases with larger reachability sets.

**The Maximal Cover Policy (MCOV):** The size of the coverage set of a case can also be used as a measure of case usefulness. Cases with large coverage sets can be used to classify many target cases and as such must make a significant contribution to classification competence. These cases should be preserved in any edited set. The *maximal cover* (MCOV) evaluation function implements this idea by biasing cases with large coverage sets.

**The Relative Cover Policy (RC):** The RFC and MCOV functions, and indeed the ICF method discussed in the previous section, use local measures of case competence as an editing guide: coverage and reachability sets encode local competence characteristics only. These sets tell us very little about the global competence characteristics of cases. For example, a case c may have a large coverage set, and according to MCOV it would be a prime candidate for inclusion in the edited set. However, if the cases that c covers are themselves covered by other cases then the unique competence contribution of c is reduced. What is needed is a metric for computing the competence of a case relative to other cases. Relative coverage is just such a metric (3). It is based on the idea that if a case c is covered by n other cases then each of the n cases will receive a contribution of 1/n from c to their relative coverage measures [11]. The RC evaluation function uses the relative coverage metric as a measure of case usefulness by preferring cases with higher relative coverage values.

$$RelCov(c) = \sum_{c' \in CoverageSet(c)} \frac{1}{|ReachabilitySet(c')|} \quad (3)$$

### 3.2 Addition Rule

We propose two possible addition rules. The first, and simplest, is the *null addition rule*. According to this rule every case considered during editing is added to the edited set. This still produces an edited set of cases when combined with the coverage deletion rule to remove cases from the training set (see Section 3.3).

The second addition rule is based on the CNN rule. Cases are only added to the edited set if they cannot be correctly classified by this set. This rule will produce smaller edited sets than the null addition rule, but it is not clear what sort of impact this rule will have on the accuracy of the resulting edited set (see Section 4).

### 3.3 Deletion Rule

During each iteration of editing, after a case has been considered for addition to the edited set, there is an opportunity to remove cases from the remaining training set. We propose two possible deletion rules. The simplest is the *null deletion rule*, which only deletes the current case from the remaining training set. A second option, the *coverage deletion rule*, deletes all cases that the current case can be used to classify (that is, all cases that the current case covers) on the grounds that these cases must be redundant in the context of the current edited set. Eagerly deleting groups of cases from the remaining training set during each round of editing will lead to the production of smaller edited sets but there may be a corresponding decrease on classification accuracy.

### 3.4 Update Rule

Our editing algorithms are guided by a competence model, the coverage and reachability sets of the training cases. These sets are the basis for the evaluation functions and the coverage deletion rule. The model is initialised with respect to the entire set of original training cases. As cases are removed from the training set there is an opportunity to recompute the model with respect to the remaining training cases. The update parameter specifies whether or not the competence model is re-initialised after each iteration.

The evaluations of cases can change due to the absence of cases from the training set, and this may lead to the addition of redundant cases to the edited set. The advantage of updating the competence model during editing is that it ensures that the evaluation functions used to prioritise the remaining cases are optimised with respect to the current set of remaining cases. Updating the model after each iteration should result in smaller edited sets but the accuracy of these sets may be compromised due to overfitting in the competence model - each new version of the competence model is computed with respect to a smaller set of training cases. These cases may no longer be representative of future target problems.

### 3.5 The Family Tree

This family contains 24 individual algorithms, since there are 24 different combinations of ordering strategy (x3), addition rule (x2), deletion rule (x2), and update rule (x2). Six of these combinations can be ignored since they do not produce edited sets that are smaller than the original training cases. This occurs in those algorithms that do not use the CNN addition rule or the coverage deletion rule (combinations, FFF, FFT for MCOV, RFC, and RC). This leaves a total of 18 different editing algorithms.

## 4 Evaluation

In this section we evaluate the performance of our new family of competence-guided editing techniques on a range of data sets and by comparison with a number of existing editing techniques.

### 4.1 Experimental Set-Up

We choose 11 classification datasets from the UCI ML Repository [2]. For each data set a random 20% of the instances are held back as unseen target instances. This is repeated 30 times, to produce 30 training and test sets for each data set. We also choose three editing methods to compare against our new techniques: (1) CNN - to facilitate a comparison with a pure redundancy reduction technique; (2) RENN - to facilitate a comparison with a pure noise removal technique; and (3) ICF - to facilitate a comparison with a state-of-the-art hybrid method; ICF is chosen because it is directly related to our techniques (in its use of a similar competence model), and because recent results suggest it is one of the leading editing algorithms currently available [4].

### 4.2 Methods and Results

For each data set, we compute the classification accuracies (as a percentage of correct classifications) and edited set sizes (as a percentage of the training set) produced by each of the editing algorithm. This gives 30 accuracy and size values for each data set and algorithm pairing, from which we compute a mean values.

We also compute benchmark accuracy values for each data set by classifying each test set by using the corresponding training sets without editing. The benchmark size for each data set is 100%.

The mean accuracy and size for the 11 data sets, and the 21 algorithmic variations, are shown in Figure 4; each table cell holds two values, the top one is the edited set size as a percentage of the original training set, and the bottom one is the percentage accuracy. Mean values for each algorithm over all data sets are also shown.

### 4.3 Analysis

In general, these results are extremely positive. All of the new competence-guided techniques produce significantly smaller edited sets than any other technique, while maintaining comparable accuracy. In fact, 6 algorithms from our new family out-perform ICF and CNN in terms of edited size and accuracy.

#### *4.3.1 General Performance*

Figure 2 documents the best performing of our new algorithms; we present the MCOV, RFC, and RC variants that produce the *smallest* edited sets and those that produce the *best* classification accuracy, alongside our comparison algorithms and the benchmark results. While the smallest variants suffer from some reduction in classification accuracy, the best variants consistently out perform CNN and ICF. In fact, compared to ICF and CNN, the overall best of our new family, algorithm 6 (RC with the FTF combination), delivers improved classification accuracy with less than 50% of the cases needed by the ICF edited set and less than 20% of the cases needed by CNN. This best new algorithm needs only 7.35% of the original training cases to achieve 77.79% accuracy (see Figure 4).

#### *4.3.2 MCOV vs. RFC vs. RC*

Figure 3 graphs the mean performance (size and accuracy) of our 18 family members over all data sets; the top set of curves corresponds to accuracy. From this graph we can clearly see the impact of different parameter combinations on the size and accuracy of the resulting edited sets. In general, the 6 MCOV algorithms produce the smallest edited sets, with an average size of 5.22%, as compared to 6.79% and 5.53% for the RFC and RC variants, respectively. However, the MCOV algorithms also suffer the greatest accuracy degradation. They produce edited sets with a mean accuracy of 74.96%, compared to 75.82% and 76.52% for the RFC and RC variants, respectively. These results seem to confirm our earlier hypothesis that the use of non-local competence measures such as relative coverage

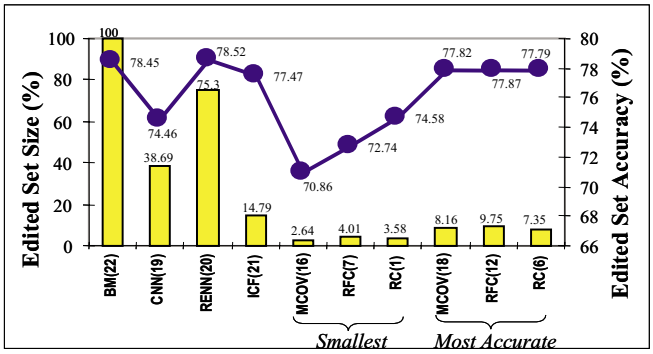


**Figure 2.** Comparative performance results for the best performing (smallest and most accurate) of the new competence-guided techniques. The bar-graph represents the size results and the line-graph represents the accuracy results.

should produce superior editing techniques. The RC algorithms produce edited sets that are not significantly larger than the MCOV sets, and yet the RC sets benefit from significantly improved accuracy.

### *4.3.3 Parametric Variations*

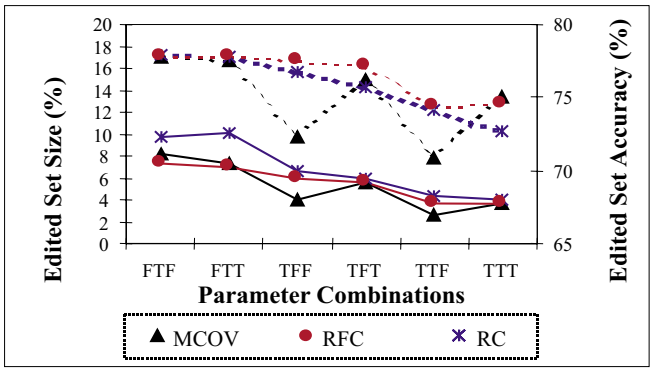


**Figure 3.** Performance results for the different parametric variations of the competence-guided techniques; the accuracy results are indicated by dotted lines, and the size results by solid lines.

Figure 3 also provides information about the impact of the different parametric variations for each algorithm; that is, the addition rule, deletion rule, and update rule variations. In terms of edited set size, the CNN addition rule is seen to produce the greatest size reductions, followed by the coverage deletion rule. In general, the TTF combination (CNN addition rule, coverage deletion rule, and no model update) produces the smallest edited sets. Similarly, the FTF combination (null addition rule, coverage deletion rule, and no model update) produces the largest edited sets (a mean size of 8.42% across MCOV, RFC, and RC). This is consistent with our earlier predictions (see Section 3).

Similar observations can be made regarding edited set accuracy, although this is largely due to the inherent dependency between edited set size and accuracy. The TTF combination tends to produce edited sets that suffer from a significant loss in accuracy; a mean accuracy for the TTF combination of 73.12% is observed across the MCOV, RFC, and RC variants. Similarly, the FTF combination produces edited sets with a mean accuracy of 77.82%.

To sum up: this new editing family offers a range of algorithms with excellent performance characteristics, and individual members consistently out-performing common techniques, such as CNN and ICF, in both edited set size and accuracy.

## 5 Conclusions

Training set editing is a core issue for improving the cost and accuracy of many lazy learning techniques including case-based learning, instance-based learning, memory-based reasoning, and nearest-neighbour methods. Traditional editing techniques utilise one of two basic strategies: (1) add selected cases to a growing edited set, from the original training set; or, (2) delete selected cases from the original training set to produce a filtered edited set.

We have described a new family of techniques that employ both of these strategies, and that are guided by an explicit model of classification competence. We have demonstrated that, as a whole, this family of algorithms benefits from superior performance characteristics, and that individual members significantly out-perform traditional and state-of-the-art editing techniques in terms of edited set size and accuracy.

## REFERENCES


[1] D.W. Aha, D. Kibler, and M.K. Albert, 'Instance-Based Learning Algorithms', *Machine Learning*, **6**, 37–66, (1991).

[2] C. Blake, E. Keogh, and C.J. Merz, *UCI Repository of Machine Learning Algorithms Databases*, Irvine, CA: University of California. Department of Information and Computer Science, 1998.

[3] H. Brighton, *Information Filtering for Lazy Learning Algorithms*, Masters thesis, Centre for Cognitive Science, University of Edinburgh, Scotland, 1997.

[4] H. Brighton and C. Mellish, 'On the Consistency of information filters for Lazy learning algorithms', in *Proceedings of the 3rd European Conference on the Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science*, ed., Jan Rauch Jan M. Zytkow, pp. 283–288. Springer Verlag, (1999).

[5] C.L Chang, 'Finding Prototypes for Nearest Neighbour Classifiers', *IEEE Transactions on Computers*, **C-23**, 1179–1184, (1974).

[6] G.W. Gates, 'The Reduced Nearest Neighbor Rule', *IEEE Transactions on Information Theory*, **IT-18(3)**, 431–433, (1972).

[7] P.E. Hart, 'The Condensed Nearest Neighbor Rule', *IEEE Transactions on Information Theory*, **IT-14**, 515–516, (1967).

[8] D. Kibler and D.W Aha, 'Learning representative exemplars of concepts: An initial case study.', in *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 24–30. Morgan Kaufmann, (1987).

[9] B. Smyth and M.T. Keane, 'Remembering to Forget: A Competence Preserving Case Deletion Policy for CBR Systems', in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, ed., Chris Mellish, pp. 377–382. Morgan Kaufmann, (1995).

[10] B. Smyth and E. McKenna, 'Modelling the Competence of Case-Bases', in *Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence*, eds., B. Smyth and P. Cunningham, pp. 208–220. Springer Verlag, (1998).

[11] B. Smyth and E. McKenna, 'Building Compact Competent Case-Bases', in *Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence*, eds., Klaus Dieter Althoff, Ralph Bergmann, and L.Karl Branting, pp. 329–342. Springer Verlag, (1999).

[12] I. Tomek, 'An Experiment with the Edited Nearest-Neighor Rule', *IEEE Transactions on Systems, Man, and Cybernetics*, **6(6)**, 448–452, (1976).

[13] I. Tomek, 'Two Modifications of CNN', *IEEE Transactions on Systems, Man, and Cybernetics*, **7(2)**, 679–772, (1976).

[14] D.L. Wilson, 'Asymptotic Properties of Nearest Neighbor Rules Using Edited Data', *IEEE Transactions on Systems, Man, and Cybernetics*, **2-3**, 408–421, (1972).

[15] D.R. Wilson and T.R. Martinez, 'Instance Pruning Techniques', in *Proceedings of the 14th International Conference on Machine Learning*, ed., D. Fisher, pp. 403–411. Morgan Kaufmann, (1997).

| Algorithm /Data Set | Type | Adult | BCancer | Bupa | Credit | Cylinder | Diabetes | Hepatitis | Hungarian | Ionosphere | Voting | TicTacToe | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ----RC---- | 4.94 | 0.95 | 5.15 | 3.7 | 5.32 | 4.45 | 2.93 | 3.71 | 2.76 | 1.67 | 3.82 | **3.58** |
| | TTT | 72.64 | 94.52 | 61.64 | 78.07 | 66.08 | 60.89 | 80.11 | 75.29 | 74.19 | 88.43 | 68.53 | **74.58** |
| 2 | TFF | 6.06 | 1.48 | 7.79 | 7.21 | 8.16 | 6.78 | 3.74 | 5.48 | 3.81 | 2.84 | 11.30 | **5.87** |
| | | 74.67 | 95.68 | 63.62 | 81.76 | 68.89 | 62.72 | 81.83 | 77.99 | 75.05 | 90.93 | 79.65 | **77.51** |
| 3 | TFT | 5.67 | 1.34 | 7.22 | 6.59 | 7.96 | 6.50 | 3.58 | 5.07 | 3.85 | 2.97 | 11.15 | **5.63** |
| | | 74.47 | 95.61 | 63.82 | 81.06 | 68.33 | 59.78 | 81.51 | 78.44 | 75.14 | 91.46 | 78.87 | **77.14** |
| 4 | TTF | 4.92 | 0.89 | 5.19 | 4.11 | 5.59 | 4.45 | 2.98 | 3.74 | 2.81 | 1.71 | 4.60 | **3.73** |
| | | 71.14 | 94.52 | 61.88 | 78.67 | 66.36 | 60.06 | 79.78 | 75.34 | 73.81 | 85.59 | 71.03 | **74.38** |
| 5 | FTT | 6.35 | 2.12 | 9.88 | 10.68 | 11.20 | 8.28 | 4.41 | 7.40 | 4.67 | 4.14 | 7.94 | **7.01** |
| | | 75.28 | 95.83 | 64.64 | 83.26 | 70.80 | 61.11 | 81.72 | 79.60 | 73.52 | 91.57 | 77.71 | **77.73** |
| 6 | FTF | 6.61 | 2.20 | 10.06 | 11.23 | 11.63 | 8.39 | 4.57 | 7.49 | 4.71 | 4.46 | 9.50 | **7.35** |
| | | 75.65 | 96.06 | 65.27 | 83.24 | 70.99 | 60.94 | 81.61 | 79.66 | 73.62 | 91.76 | 76.89 | **77.79** |
| 7 | ---RFC--- | 5.29 | 1.11 | 5.79 | 4.31 | 5.95 | 5.23 | 4.36 | 3.63 | 4.98 | 2.17 | 1.30 | **4.01** |
| | TTT | 74.07 | 91.76 | 62.37 | 76.84 | 66.23 | 60.89 | 77.74 | 68.56 | 69.14 | 86.97 | 65.60 | **72.74** |
| 8 | TFF | 7.06 | 1.91 | 8.64 | 7.49 | 9.17 | 7.23 | 5.75 | 6.12 | 6.71 | 3.80 | 8.29 | **6.56** |
| | | 75.89 | 95.76 | 64.11 | 80.89 | 69.94 | 60.50 | 79.25 | 78.97 | 72.43 | 91.11 | 75.08 | **76.72** |
| 9 | TFT | 7.10 | 1.88 | 8.16 | 7.16 | 8.90 | 7.05 | 5.70 | 6.07 | 6.52 | 3.78 | 2.85 | **5.93** |
| | | 75.73 | 95.63 | 63.09 | 80.51 | 69.35 | 60.11 | 79.25 | 79.02 | 71.95 | 90.81 | 67.35 | **75.71** |
| 10 | TTF | 5.32 | 1.32 | 6.14 | 4.77 | 6.27 | 5.72 | 4.68 | 4.04 | 4.80 | 2.20 | 3.42 | **4.43** |
| | | 73.74 | 94.57 | 62.46 | 79.32 | 67.01 | 60.33 | 78.50 | 73.28 | 69.67 | 87.20 | 69.42 | **74.14** |
| 11 | FTT | 12.02 | 2.63 | 12.38 | 13.83 | 13.67 | 10.10 | 8.28 | 9.10 | 10.21 | 5.56 | 13.14 | **10.08** |
| | | 75.61 | 96.01 | 63.38 | 82.73 | 71.85 | 60.83 | 79.68 | 79.89 | 74.52 | 90.84 | 79.91 | **77.75** |
| 12 | FTF | 11.57 | 2.55 | 12.11 | 13.24 | 13.32 | 9.92 | 8.23 | 8.67 | 9.96 | 5.44 | 12.20 | **9.75** |
| | | 75.49 | 95.93 | 63.29 | 82.56 | 71.51 | 60.83 | 79.79 | 80.34 | 75.67 | 91.38 | 79.77 | **77.87** |
| 13 | -MCOV- | 4.67 | 0.68 | 5.27 | 4.07 | 5.61 | 4.55 | 2.82 | 3.84 | 2.95 | 1.87 | 4.03 | **3.67** |
| | TTT | 70.81 | 95.61 | 62.17 | 78.65 | 67.16 | 60.94 | 79.68 | 77.41 | 71.48 | 91.72 | 69.98 | **75.06** |
| 14 | TFF | 5.02 | 0.75 | 6.1 | 4.61 | 7.39 | 5.16 | 2.58 | 4.25 | 2.50 | 2.39 | 3.09 | **3.98** |
| | | 65.53 | 96.01 | 61.40 | 76.45 | 67.59 | 57.50 | 76.88 | 75.75 | 72.38 | 90.35 | 54.90 | **72.25** |
| 15 | TFT | 6.36 | 0.97 | 7.61 | 6.48 | 8.31 | 6.79 | 3.60 | 4.79 | 3.15 | 2.94 | 9.85 | **5.53** |
| | | 73.54 | 95.76 | 63.77 | 79.76 | 69.97 | 58.83 | 78.71 | 78.51 | 71.90 | 90.54 | 77.68 | **76.27** |
| 16 | TTF | 3.63 | 0.54 | 4.26 | 2.72 | 4.73 | 3.88 | 1.91 | 2.63 | 1.89 | 1.67 | 1.15 | **2.64** |
| | | 63.13 | 95.96 | 58.41 | 74.76 | 65.52 | 59.00 | 75.91 | 73.51 | 72.33 | 90.81 | 50.16 | **70.86** |
| 17 | FTT | 6.49 | 2.34 | 10.46 | 11.33 | 11.68 | 8.66 | 4.73 | 7.59 | 4.70 | 4.25 | 8.92 | **7.38** |
| | | 75.45 | 95.51 | 64.25 | 83.50 | 72.04 | 60.56 | 81.29 | 78.79 | 72.43 | 91.38 | 77.85 | **77.55** |
| 18 | FTF | 7.11 | 2.68 | 10.99 | 12.77 | 12.90 | 9.66 | 5.03 | 8.02 | 4.83 | 5.08 | 10.67 | **8.16** |
| | | 76.1 | 95.81 | 64.35 | 83.45 | 71.33 | 60.33 | 81.72 | 79.31 | 73.05 | 92.26 | 78.32 | **77.82** |
| 19 | CNN | 42.85 | 11.67 | 57.03 | 38.64 | 44.07 | 59.03 | 39.81 | 37.84 | 43.29 | 16.88 | 34.50 | **38.69** |
| | | 71.91 | 92.88 | 61.88 | 76.69 | 75.56 | 60.06 | 73.98 | 75.00 | 69.05 | 90.65 | 82.39 | **75.46** |
| 20 | RENN | 73.71 | 96.36 | 58.37 | 83.74 | 69.51 | 56.53 | 74.41 | 78.36 | 71.27 | 91.05 | 75.00 | **75.30** |
| | | 77.28 | 96.75 | 63.57 | 84.88 | 71.70 | 59.83 | 81.08 | 81.72 | 73.67 | 92.57 | 80.65 | **78.52** |
| 21 | ICF | 13.16 | 3.33 | 15.98 | 13.85 | 17.79 | 14.29 | 13.06 | 8.50 | 10.40 | 9.73 | 42.55 | **14.79** |
| | | 75.77 | 95.73 | 63.53 | 83.24 | 70.90 | 59.22 | 80.54 | 79.02 | 73.95 | 91.69 | 78.64 | **77.47** |
| 22 | Benchmark | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | **100** |
| | | 75.69 | 95.01 | 62.90 | 82.39 | 79.51 | 60.78 | 77.85 | 78.16 | 76.33 | 91.61 | 82.74 | **78.45** |

**Figure 4.** Size and accuracy results for all data sets and algorithms. Each cell contains a percentage size value (top) and a percentage accuracy value (bottom). The final column contains the mean size and accuracy values for each algorithm over all data sets. Benchmark values are listed in the final row.