An Efficient Random Decision Tree Algorithm for Case-Based Reasoning Systems

Tor Gunnar Houeland
Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
houeland@idi.ntnu.no

Abstract

We present an efficient random decision tree algorithm for case-based reasoning systems. We combine this algorithm with a simple similarity measure based on domain knowledge to create a stronger hybrid algorithm. This combination is based on our general approach for combining lazy and eager learning methods. We evaluate the resulting algorithms on a case base of patient records in a palliative care domain. Our hybrid algorithm consistently produces a lower average error than the base algorithms.

Introduction

Lazy learning approaches do not draw conclusions until it is necessary, allowing them to collect all available information before doing any generalization. This has the potential advantage of including highly relevant information that an eager approach would not have access to, and adapting the reasoning to the particular characteristics of the problem query to solve. The drawback is that the system's reasoning (and computations) will only be performed all at the very end when an answer is required. Eager methods have the advantage that parts of their reasoning can be precomputed during training, and they only need to store abstract generalizations which can typically take only a fraction of the storage space.

In this paper we examine a hybrid approach that uses a modified version of an eager method (random decision trees) that can be partially precomputed and partially adapted to the particular problem query. We examine the results for determining similarity in a data set describing the results of palliative care for cancer patients, which is an ongoing topic of investigation in our research group.

For our random decision tree (RDT) algorithm the forest of random trees can be grown once before the system begins reasoning about cases, and we use internal data structures that can be incrementally updated in an efficient manner when new cases are added to the case base. The data structures additionally support efficiently considering only selected subsets of the case base as training data at runtime. This capability is combined with a simple similarity measure based on domain knowledge to simulate having trained the algorithm on only the most relevant cases in a computationally inexpensive manner. This high computational efficiency and ability to be integrated with other uses of cases are the primary strengths of our RDT algorithm.

In the next section we summarize some earlier research results and relate it to our work. This is followed by a description of our RDT-based experiment in the domain of palliative care for cancer patients. We describe and compare 4 different algorithms and discuss empirical results for running the algorithms on a case base of palliative care patients. Concluding remarks end the paper.

Related Research

The topic of indexing cases beforehand to support efficient retrieval during problem solving has been extensively studied in the literature. A popular form of indexing structure is a tree with cases at the leaf nodes. One early example of such a tree-based indexing structure used in case-base reasoning is a kd-tree (Wess, Althoff, and Derwand 1994), which partitions the indexing space into disjoint areas. Each leaf node is called a $bucket$, and contains the cases within a particular area of the indexing space.

An ensemble method combines multiple models to produce a better result than any individual model would. This approach has also been used for indexing trees, where multiple trees are created and combined to address a single problem. Perhaps the most well-known example of such a tree ensemble is the Random Forest (RF) classifier (Breiman 2001). RF grows a number of decision trees based on bootstrap samples of the training data. For each node of a tree, m variables are randomly chosen and the best split based on these m variables is calculated based on the bootstrap data. Each decision tree results in a classification and is said to cast a vote for that classification, and the ensemble classifier returns the class that received the most votes. RF can also compute $proximities$ between pairs of cases that can be used for clustering and data visualization, and as a similarity measure for case-based reasoning. We use a similar concept of $proximity$ to measure similarity in our random decision tree algorithm.

Diversity is an important aspect of an ensemble classifier that affects its accuracy. Bian and Wang (2007) found that the performance of an ensemble learning approach varied considerably for different applications. They studied homo-

geneous and heterogeneous ensembles and found connections between diversity and performance, and an increased diversity for heterogeneous ensembles.

Gashler et. al. (2008) examined ways to increase the diversity for ensembles of decision trees. They compared an approach that split on randomly selected attributes as a means to increase diversity, with an approach that combined multiple tree algorithms to increase diversity. For our random decision tree algorithm we similarly use entirely random attributes, and also perform the splits at random.

Ferguson and Bridge (2000) describe a generalization of similarity measures they call similarity metrics, and build upon a way of combining similarity metrics called prioritization. This approach uses an indifference relation to prioritize a secondary metric when no significant difference is found based on the primary similarity metric. In our approach we also use two similarity measurements, but they are integrated in a hybrid combination.

Richter (1995) introduced the knowledge container model for CBR systems. In this model the system's knowledge is categorized into four categories: case vocabulary, cases, similarity assessment and adaptation knowledge. A distinction was also made between compiled knowledge, which is "compiled" in a very general sense before actual problem solving begins, and interpreted knowledge that is assessed at run time, during the process of problem solving. In our research we integrate the similarity assessment with the internal storage of cases, and develop a method for efficiently "compiling" the similarity knowledge for an ensemble of random decision trees.

The utility problem occurs when additional knowledge learned decreases a reasoning system's performance instead of increasing it (Minton 1990; Smyth and Cunningham 1996). Theoretically this will always occur for a CBR system when the system's case base increases without bound, and has therefore been the subject of considerable research, since the problem remains relevant even as computers become faster and cheaper.

Patterson et. al. (2003) examine efficient retrieval as a means to reduce the effects of the utility problem. They present two indexing approaches that give efficiency gains of up to 20 times faster retrieval, with a small reduction in case base competency. As a similar trade-off, our research focuses on a highly efficient method for lazy reasoning, with a limitation on the type of decisions that can be performed to determine similarity.

In an earlier study (Houeland and Aamodt 2010), we suggest that the usefulness of an optimization should be measured by the effect it has on the reasoning system's overall utility. We continue this line of reasoning in the research presented here, by examining the trade-off between accuracy and speed that occurs in the developed algorithms.

Random Decision Tree Experiment

Our random decision tree (RDT) algorithm is an example of a general approach to combining machine learning methods with case-based reasoning. Like a traditional CBR approach we retrieve the local cases nearest to our input query (shown in figure 1), but we retrieve an unconventionally large number of cases (in the following experiment we retrieve half the case base). We run a machine learning algorithm on this subset of cases as training data, partially combining the lazy and local attributes of a CBR retrieval with the eager and global methods often used for more traditional machine learning algorithms.

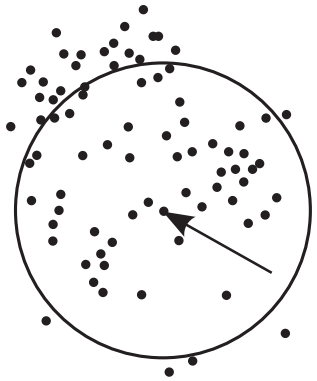

Figure 1: Selecting a similarity-based local case subset for use as training data. The cases within the circle are retrieved as the closest neighbors of the marked case, and are used to train an independent learning algorithm.

This combination is based on the intuition that CBR similarity measures can often easily express rough case relevance estimates based on domain knowledge, while eager machine learning algorithms are typically very effective at selecting precisely the best option among possible alternatives, but without the beneficial features of laziness.

A drawback of this approach is that a straightforward application of an eager method would have to be trained from scratch on the case subset during problem solving. This can be prohibitively expensive because eager methods typically perform a lot of computations that are normally amortized over many subsequent problem solving sessions.

We develop an RDT variant where the storage of the tree knowledge is inverted, by associating each case with its result for every tree and storing the knowledge with the cases. This is in contrast to the more traditional approach of creating the tree structure based on the training data and implicitly storing the knowledge in the trees. Our variant allows very efficient incremental updates for additional training data, i.e. learning new cases one by one.

In this experiment we use a forest of randomly grown trees to determine the similarity between two cases. Each tree is a fully grown binary tree of height 5, where one particular measurement is compared to a threshold value at each node.

An example tree generated by our implementation is shown in figure 2. The root node "Addiction $<$ 0.5739" in the example compares the patient's *addiction* value to the specified threshold 0.5739, and continues down the left branch marked with a dotted line in the figure if it's below the threshold value and otherwise down the right branch marked with a solid line.

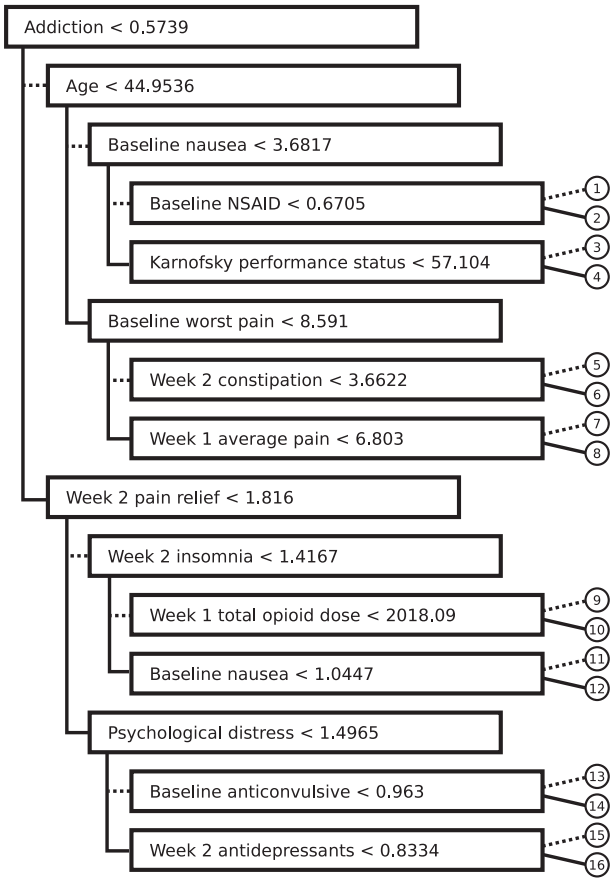

Figure 2: A randomly generated decision tree for the palliative care domain.

Each tree sorts a case into one of 16 leaf node $buckets$, or no bucket if a node would compare a measurement the case does not include (because it is a partial patient record). The forest of trees acts as a measure of similarity between cases, where two cases are said to be more similar if they're in the same bucket for a higher number of trees.

This is the same as the $proximity$ value uses in the Random Forest (RF) classifier, but with a different algorithm for growing trees. While the RF classifier generates strong trees based on the training data, we generate completely random trees.

This aspect is more similar to Random Decision Trees (Fan et al. 2003) and the Max-diverse Ensemble (Liu, Ting, and Fan 2005) algorithm, which was shown to have nearly comparable accuracy to RF but without using bootstrap samples based on the training set.

The important advantage of completely random trees for our experiment is that growing the trees does not depend on the training data, which means that a single forest can be generated once and used for case bases with different sets of case instances. This allows us to keep the same precomputed trees when adding a case to the case base, and only incrementally update the data structures that are used to represent the cases.

Algorithms

For each new case we go through all the trees and compute which $bucket$ the case belongs to, and for each case we store these computed bucket values for each tree. The advantage of this representation is that the proximity of two cases can be computed by only iterating through the stored bucket values which can be implemented efficiently.

For our experiments we are interested in combining this knowledge-lean random decision forest method with a simple CBR-like similarity measurement based on variables medical doctors consider relevant for pain classification. This similarity measurement is the sum of the normalized differences of patients' *pain intensity*, *breakthrough pain*, *pain mechanism*, *psychological distress*, *cognitive functioning*, *addiction* and *pain localization*.

We compare 4 different similarity methods: the random decision forest, the least difference in relevant variables, a completely random approach, and a hybrid approach based on the random decision forest plus the differences in relevant variables.

Unfortunately the European Palliative Care Research Collaborative (EPCRC) has not been able to reach a consensus on how to classify pain, so there is no clear "solution" or "answer" for our cases. We have decided to estimate the similarity by comparing the difference in *worst pain* and *average pain* experienced after three weeks and basing our comparisons only on the data acquired before three weeks had passed.

Our data set consists of 1486 cases with 55 numerical features from the first two weeks as the problem description, and these two pain classifications as the solution. This is relevant because the main palliative treatment is started during week 2 and the pain experienced afterwards depends on the treatment and is of utmost importance for a patient receiving palliative care. Our presented research uses the difference in pain levels as a means to indirectly estimate the correctness of the computed similarity.

We evaluate a similarity by measuring this combined difference in *worst pain* and *average pain* between the input query and the case in the case base that was determined to be the most similar, in effect measuring the performance of using the similarity measure for a 1-NN classifier. We simulated problem solving by going through each of the patient cases in order, attempting to solve them using the cases in the case base so far and then adding the case to the case base.

The observed results are sensitive to both random chance and the order of the cases in the case base. To achieve a fair comparison we generated 100 different versions of input where the order of the patient data cases had been randomly shuffled, and used the same set of 100 input orders for each similarity method. We compared the average measured difference for all problem solving attempts for the 100 different orders between the different similarity methods (excluding the first case in each order, when the case base is empty). This approach was used to empirically evaluate 4 different similarity methods for our domain, which are presented in increasing order of complexity.

Algorithm 1 RANDOM-SIMILARITY

```
1.  Cases ← EMPTY
2.  for each patient p ∈ PATIENTS
3.    do q ← CREATE-INPUT-QUERY(p)
4.       x ← RANDOM-INTEGER(1, length[Cases])
5.       best-case ← cases[x]
6.       ▷ Use best-case as the solution for query q
7.       APPEND(Cases, p)
```

RANDOM-SIMILARITY simply chooses a random case from the case base and naturally gives the worst results and an average error of about 4.98, but is a baseline to work against that signifies no correlation to real similarity. Like the other algorithms it is used to select a case *best-case* as the solution (line 6) in a case-based reasoning system, which is afterwards learned and added to the case base before the next query is received. Line 3 extracts the problem description for a case, which in our domain is the patient data from the first two weeks.

Algorithm 2 LEAST-DIFFERENCE

```
1.  Cases ← EMPTY
2.  for each patient p ∈ PATIENTS
3.    do q ← CREATE-INPUT-QUERY(p)
4.       closest ← ∞
5.       best-case ← NIL
6.       for each case c ∈ Cases
7.         do diff ← CBR-DIFFERENCE-MEASURE(q, c)
8.            if diff < closest
9.              then closest ← diff
10.                  best-case ← c
11.      ▷ Use best-case as the solution for query q
12.      APPEND(Cases, p)
```

LEAST-DIFFERENCE is a straight-forward CBR system that uses the simple CBR-DIFFERENCE-MEASURE based on the normalized differences in the 7 variables a medical doctor expected to be relevant. Using this method as essentially a 1-NN classifier gives an average error of about 4.55.

Algorithm 3 N-RANDOM-TREES

```
1.  Cases ← EMPTY
2.  Buckets ← EMPTY
3.  Trees ← GENERATE-TREES(N)
4.  for each patient p ∈ PATIENTS
5.    do q ← CREATE-INPUT-QUERY(p)
6.       Buckets[q] ← COMPUTE-TREE-BUCKETS(q, Trees)
7.       most-similar ← (−∞)
8.       best-case ← NIL
9.       for each case c ∈ Cases
10.        do sim ← COMPUTE-PROXIMITY(Buckets[q],
           Buckets[c])
11.           if sim > most-similar
12.             then most-similar ← sim
13.                  best-case ← c
14.      ▷ Use best-case as the solution for query q
15.      APPEND(Cases, p)
```

N-RANDOM-TREES is based on our approach for efficiently using random decision trees in a CBR system. The GENERATE-TREES procedure builds N fully grown binary trees of height 5, which is stored as an array of 15 *attribute-threshold* pairs that represent the nodes in the tree. When generating a tree, the attribute to compare at each node is selected randomly, and the threshold is set to a random value chosen from that attribute's range of possible values. For our experiment we do not have the domain knowledge to determine the true distribution of possible attribute values. For simplicity we choose to select uniformly random values between the highest and lowest values that are contained in the data set.

The COMPUTE-TREE-BUCKETS procedure computes the resulting $bucket$ for each of the N trees and returns the previously described N-value representation that is used for efficient comparisons. The COMPUTE-PROXIMITY procedure iterates through the bucket values for two cases and returns the number of matching values, i.e. the number of trees where the two cases end up in the same leaf node.

Algorithm 4 N-HYBRID

```
1.  Cases ← EMPTY
2.  Buckets ← EMPTY
3.  Trees ← GENERATE-TREES(N)
4.  for each patient p ∈ PATIENTS
5.    do q ← CREATE-INPUT-QUERY(p)
6.       Buckets[q] ← COMPUTE-TREE-BUCKETS(q, Trees)
7.       Distance ← EMPTY
8.       for each case c ∈ Cases
9.         do Distance[c] ← CBR-DIFFERENCE-
           MEASURE(q, c)
10.      Closest-Cases ← The closest half of Cases sorted
         according to Distance
11.      most-similar ← (−∞)
12.      best-case ← NIL
13.      for each case c ∈ Closest-Cases
14.        do sim ← COMPUTE-PROXIMITY(Buckets[q],
           Buckets[c])
15.           if sim > most-similar
16.             then most-similar ← sim
17.                  best-case ← c
18.      ▷ Use best-case as the solution for query q
19.      APPEND(Cases, p)
```

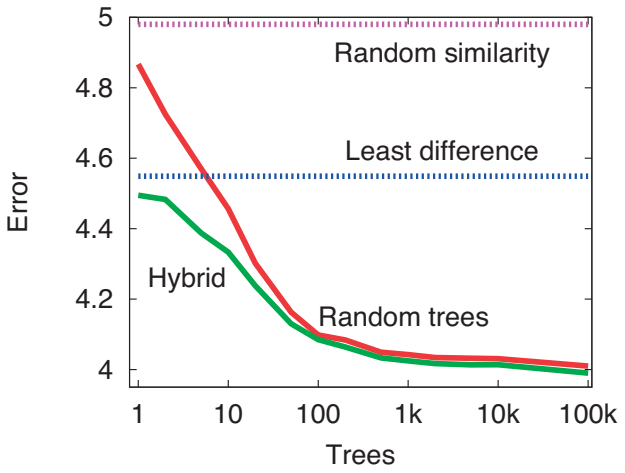

Figure 3: Measured error in the palliative care domain compared to the number of trees for our 4 algorithms. (Lower error is better.)

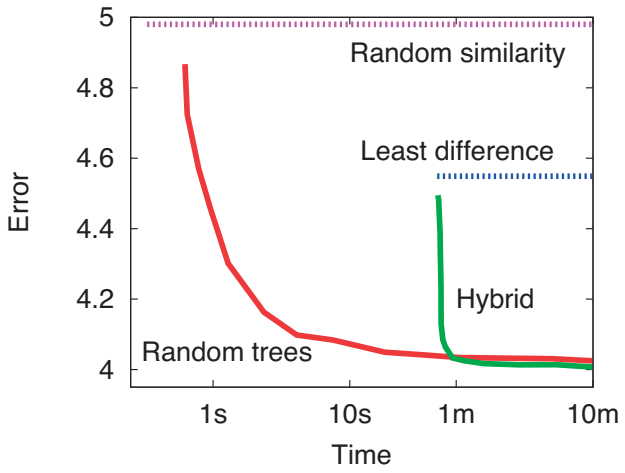

Figure 4: Measured error in the palliative care domain compared to computational time required for our 4 algorithms. (Lower error is better.)

N-HYBRID is a hybrid combination of the LEAST-DIFFERENCE approach and our efficient random decision tree implementation. For a given problem query it considers only the closest half of the case base, as measured by the CBR-DIFFERENCE-MEASURE function. This hybrid approach is proposed as a general way to combine similarity measures based on domain knowledge with knowledge-lean methods, and can be straight-forwardly adapted to ensembles of classification trees.

In our domain, the CBR-DIFFERENCE-MEASURE acts as a domain knowledge-based guard against spurious similarities detected by the random trees. Our random forest implementation performs an unguided similarity comparison, without considering whether that similarity applies for classifying pain or not. The pain treatment domain knowledge present in the CBR-DIFFERENCE-MEASURE function complements the ”raw” similarity computed by the trees, combining the two diverse knowledge sources in a CBR system in a way that has some similarities to the approaches used for very heterogeneous combinations in ensemble classifiers. Just as for ensemble classifiers these combinations do not necessarily improve performance for CBR systems if the knowledge sources do not complement each other.

In our experimental setup where the first similarity measure selects a local case subset that contains a great number of cases (half the original cases), the effect of the second similarity measure that reduces this down to a single case will be larger than the first. Because of this the combination is most successful when the stronger tree-based approach is used as the second similarity measure. It is possible to use the tree-based approach to reduce the case base and then the CBR-DIFFERENCE-MEASURE function to select the single nearest case, but this does not produce as good results for our domain, with an error of around 4.3 (depending on the number of trees). This is as expected, because our hybrid approach improves the results compared to only using the second similarity measure. For a large number of trees the error from the N-RANDOM-TREES algorithm is significantly lower than for LEAST-DIFFERENCE, and thus the starting point for the hybrid combination is better.

Results

Figure 3 shows the measured error compared to the number of trees in the forest for the 4 different algorithms. RANDOM-SIMILARITY and LEAST-DIFFERENCE do not depend on the number of trees and are shown as horizontal lines for their error level.

The error for N-RANDOM-TREES and N-HYBRID rapidly decreases for N values up to around 100 trees, and then starts flattening out, and more than 1000 trees only gives a slight decrease in error. This is expected as additional trees provide less new information when a larger proportion of possible attribute combinations have already been examined, and in general it becomes increasingly difficult to improve a result the lower the remaining error is.

The N-HYBRID algorithm consistently provides lower error than the base algorithms N-RANDOM-TREES and LEAST-DIFFERENCE, which suggests that our combined hybrid approach works successfully in our domain. For a large number of trees the errors for the N-HYBRID and N-RANDOM-TREES algorithms are relatively close in absolute value, but achieving the same incremental improvement using only a larger N value would be much more computationally expensive than using the hybrid approach, and might possibly be unachievable at the highest end as the error continues to flatten out around 4.0.

Figure 4 also shows the measured error for the 4 different algorithms, but compared according to the time (computational resources) required. (For legibility the RANDOM-SIMILARITY and LEAST-DIFFERENCE algorithms are shown as extended horizontal lines, while in reality they consist of only the leftmost point since their execution time for a given set of inputs remains constant apart from small random fluctuations in the computing environ-

ment.)

While the exact values are highly dependent on the type of computing machine it's measured on, we are interested in the relative differences between algorithms which are mostly determined by their computational complexity.

We see that the LEAST-DIFFERENCE algorithm is considerably more computationally expensive than the efficient N-RANDOM-TREES implementation for low values of N. Running the LEAST-DIFFERENCE algorithm is roughly comparable to 1000 random trees, and this overhead is reflected for the N-HYBRID algorithm as well.

The increase in computational cost to generate hundreds of trees for N-HYBRID is relatively modest compared to the cost of including the LEAST-DIFFERENCE computations at all, which means that the N-HYBRID algorithm rapidly becomes the best-performing of the 4 algorithms once enough computational resources are allotted to run it at all.

Algorithm	Time	Error
RANDOM-SIMILARITY	0.3 seconds	4.98
LEAST-DIFFERENCE	45 seconds	4.55
1-RANDOM-TREE	0.5 seconds	4.87
10-RANDOM-TREES	1 second	4.46
100-RANDOM-TREES	4 seconds	4.10
1000-RANDOM-TREES	30 seconds	4.04
10000-RANDOM-TREES	300 seconds	4.03
100000-RANDOM-TREES	3030 seconds	4.01
1-HYBRID	45 seconds	4.50
10-HYBRID	45 seconds	4.33
100-HYBRID	50 seconds	4.08
1000-HYBRID	70 seconds	4.02
10000-HYBRID	320 seconds	4.01
100000-HYBRID	2730 seconds	3.99

Table 1: Numerical results for our 4 algorithms in the palliative care domain.

Table 1 shows a more detailed numerical display of an illustrative subset of the results for power-of-10 values of N. 100-RANDOM-TREES is a very quick algorithm that produces a relatively low error compared to the other algorithms, while the best results are achieved by the N-HYBRID algorithm for high values of N. It is also interesting to note that 100000-HYBRID runs faster than 100000-RANDOM-TREES, which is because in the HYBRID version the somewhat expensive operation of comparing 100k trees is only performed for half as many cases.

Conclusions

In this paper we have developed a new random decision tree (RDT) algorithm that can be very efficiently implemented in a CBR setting. We have combined this RDT algorithm with a simple traditional similarity measure based on domain knowledge to create a hybrid similarity assessment algorithm. The hybrid combination outperformed the base algorithms it was based on by returning predictions with consistently lower error on average for cases from a palliative care domain.

Acknowledgments

We wish to thank Cinzia Brunelli for providing the data set, and Anne Kari Knudsen for interpreting the data and analysing the relevance of the features from a clinical perspective.

References

Bian, S., and Wang, W. 2007. On diversity and accuracy of homogeneous and heterogeneous ensembles. *Int. J. Hybrid Intell. Syst.* 4:103–128.

Breiman, L. 2001. Random forests. *Machine Learning* 45:5–32. 10.1023/A:1010933404324.

Fan, W.; Wang, H.; Yu, P. S.; and Ma, S. 2003. Is random model better? on its accuracy and efficiency. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, 51–. Washington, DC, USA: IEEE Computer Society.

Ferguson, A., and Bridge, D. 2000. Generalised weighting: A generic combining form for similarity metrics. In *Procs. of Eleventh Irish Conference on Artificial Intelligence & Cognitive Science, J.Griffith and C.O'Riordan*, 169–179.

Gashler, M.; Giraud-Carrier, C.; and Martinez, T. 2008. Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous. In *2008 Seventh International Conference on Machine Learning and Applications*, 900–905. IEEE.

Houeland, T., and Aamodt, A. 2010. The utility problem for lazy learners - towards a non-eager approach. In Bichindaritz, I., and Montani, S., eds., *Case-Based Reasoning. Research and Development*, volume 6176 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 141–155. 10.1007/978-3-642-14274-1.

Liu, F. T.; Ting, K. M.; and Fan, W. 2005. Maximizing tree diversity by building complete-random decision trees. In Ho, T. B.; Cheung, D.; and Liu, H., eds., *Advances in Knowledge Discovery and Data Mining*, volume 3518 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 605–610. 10.1007/11430919.

Minton, S. 1990. Quantitative results concerning the utility of explanation-based learning. *Artif. Intell.* 42(2-3):363–391.

Patterson, D. W.; Rooney, N.; and Galushka, M. 2003. Efficient retrieval for case-based reasoning. In Russell, I., and Haller, S. M., eds., *FLAIRS Conference*, 144–149. AAAI Press.

Richter, M. M. 1995. The knowledge contained in similarity measures. Invited Talk at ICCBR-95.

Smyth, B., and Cunningham, P. 1996. The utility problem analysed - a case-based reasoning perspective. In *Proceedings of the Third European Workshop on Case-Based Reasoning*, 392–399. Springer Verlag.

Wess, S.; Althoff, K.-D.; and Derwand, G. 1994. Using k-d trees to improve the retrieval step in case-based reasoning. In *Selected papers from the First European Workshop on Topics in Case-Based Reasoning*, EWCBR '93, 167–181. London, UK: Springer-Verlag.