Adaptation Knowledge from the Case Base

[illegible]

j.main@latrobe.edu.au, m.witten@latrobe.edu.au

Abstract. Case adaptation continues to be one of the more difficult aspects of case-based reasoning to automate. This paper looks at several techniques for utilising the implicit knowledge contained in a case base for case adaptation in case-based reasoning systems. The most significant of the techniques proposed are a moderately successful data mining technique and a highly successful artificial neural network technique. Their effectiveness was evaluated on a footwear design problem.

1 Case Retrieval and Case Adaptation

Case-based reasoning endeavours to solve a new problem by finding a similar past case, stored in a repository of past cases called a case base [1]. A stored case consists of the problem description and its associated solution. The problem case consists of the problem description alone. If there is an existing case with an identical problem description, its solution maps directly to become the problem case's solution. In the more likely event of a similar but not identical stored case being found, its solution needs to be modified to solve the problem case.

The first step in case-based reasoning is case retrieval [2]; given the new problem to solve, find the closest existing stored case(s). If the problem case and the retrieved case are not identical or within an acceptable level of difference, case adaptation must be carried out to modify the retrieved case's solution to take into consideration the differences between the cases' problem descriptions. The sort of adaptation required and what needs to be altered must be determined, and a strategy for performing this adaptation set in place [3]. Case adaptation continues to be one of the more difficult parts of the case-based reasoning cycle to solve in an automated fashion; that is, without the intervention or extraction of knowledge from a domain expert [4].

To achieve the worthwhile goal of automated adaptation, from where can we obtain knowledge about the relationships between a case's functional specification and its solution? This paper addresses this question. It looks at several techniques for gaining adaptation knowledge from existing cases: a relatively unsuccessful brute force technique, a moderately successful data mining technique and a highly successful technique using supervised artificial neural networks.

Several previous authors have attempted to use knowledge from the case base in adaptation. Some incorporated artificial neural networks [5][6][7][8], but our proposals are considerably different from this previous research.

1.1 Using Stored Case Information in Case Adaptation

One way of discovering knowledge in the case base is from the differences between cases in the case base. We can measure the distance between a retrieved case and a problem case by developing a distance vector of the distances between the related features in each case. For example, consider a case base of two cases, each having five input attributes and three solution components.

– Case 1 input attribute values: 1 0.5 0.8 0 1, component designs: 1A 2B 3C
– Case 2 input attribute values: 1 0.5 0.2 0 0, component designs: 1A 2B 3D

The difference between these cases' input features is the difference vector 0 0 0.6 0 1. It seems that input features 3 and 5 contribute to the design of the third solution component, as when these input features are different, there is a difference in the design of that component.

One technique that utilises difference vectors in case adaptation is, given an input case description and a retrieved case, find another difference vector between cases in the case base that is the same as the difference vector between the problem and retrieved cases. We can then determine the effect of the difference of the cases' problem features on the associated solutions for the cases by looking for differences between the solutions of the pair of cases associated with the matching input vector. Finally we apply the same transformation to the retrieved case's solution to derive the problem case's solution.

Initial proposed technique: Consider a case base of case descriptions with m problem description features and n solution description features. Let case P be a problem case with a problem case description vector $\mathbf{v}_{1p}$ (m components) and case R be a retrieved case with description vector $\mathbf{v}_{2ps}$ ($m + n$ components). If $\Delta\mathbf{v}_{p12}$ is the problem difference vector between case P and R then we seek two cases in the case base with case description vectors $\mathbf{v}_{3ps}$ and $\mathbf{v}_{4ps}$ with the same problem difference vector. We then determine the solution difference vector:

$$\Delta\mathbf{v}_{s34} = (s_{31} - s_{41}, s_{32} - s_{42}, \ldots, s_{3n} - s_{4n})$$

and propose a solution to case P by adding the solution description features of case R to $\Delta\mathbf{v}_{s34}$ giving $\mathbf{v}_{1ps}$:

$$\mathbf{v}_{1ps} = (p_{11}, p_{12}, \ldots, p_{1m}, s_{21} + s_{31} - s_{41}, s_{22} + s_{32} - s_{42}, \ldots, s_{2n} + s_{3n} - s_{4n}).$$

This initial proposal has certain failings. The sparser the case base, the less likely it is that we will find a pair of cases with an identical difference vector to the retrieved/problem case difference vector. Additionally, the more features a case has, the more combinations of features there may be in the cases in the case base. Therefore, we need to be able to search our case base for pairs of cases with similar, as well as identical, difference vectors. It may be possible to apply a type of nearest neighbour search to find the closest matching case. A similarity metric must therefore be designed for difference vectors.

Using a neural network: Perhaps it is possible to train a single neural network to discover a pair of cases with a similar difference vector? If the network inputs are mapped to a difference vector's input features and each output corresponds to a particular difference vector between a pair of cases, this could theoretically find an appropriate difference vector more quickly than the brute force search technique of nearest neighbour searching. Problems include:

1. The greater the number of cases in the case base, the greater the number of output nodes in the ANN (and hence the lower the expected proportion of 1s to 0s in the output signals). If the proportion of 1s is very low, then by all outputs converging to 0, the nodes in error that should be 1, are a small proportion of the nodes.
2. The network learns with such specificity that it essentially becomes a look-up table for content-addressable memory [9]. Thus new problem cases, unless they have the same set of input features as one of the training cases, will not be able to select among the existing difference vectors successfully.
3. The network takes an excessive amount of time to train.

Alternatives to finding a specific pair of cases with a specific difference vector are needed. Statistical data mining techniques and neural networks provide possibly more generalised ways of using the knowledge from distance vectors.

2 Using Zhou and Dillon's Symmetrical Tau

A data mining technique such as Zhou and Dillon's Symmetrical Tau [10,11] can be applied to learn the relative importance of the relationship between a particular input feature and a particular solution feature. Consider a case base of designs where each case is made up of designs for components M and N, and each case is retrieved by searching for specific values for input features X, Y and Z. The Symmetrical Tau can be used to determine whether X, Y or Z is more important in the selection of a design for component M, and similarly for component N.

This technique requires the calculation of a Tau value for each relationship between an input feature and a component of the design. It could be expected that the higher the Tau value, the more influence an input feature has on that component. For each combination of input feature and solution component, a contingency table is drawn up. For example, for the combination of input feature Y and solution component N, we would have a contingency table as in Table 1. Each example is a difference vector between two cases. The total number of examples to use in the data mining process must be chosen. Using all examples is possible but may not be necessary. Selecting a smaller number of training examples (the most similar pairs or pairs of cases at random) is generally preferable.

Using this contingency table, the Symmetrical Tau value for that relationship between input feature and solution component is defined as [10][11]:

$$\tau = \frac{\sum_{j=1}^{J}\sum_{i=1}^{I}\frac{P_{ij}^2}{P_{+j}} + \sum_{i=1}^{I}\sum_{j=1}^{J}\frac{P_{ij}^2}{P_{i+}} - \sum_{i=1}^{I}P_{i+}^2 - \sum_{j=1}^{J}P_{+j}^2}{2 - \sum_{i=1}^{I}P_{i+}^2 - \sum_{j=1}^{J}P_{+j}^2}$$

Table 1. Contingency table example

		Component N different		
		Yes	No	
Input feature Y different	Yes	A	B	E
	No	C	D	F
		G	H	

A:	no. examples with a different value for Y and a different design for component N
B:	no. examples with a different value for Y and the same design for component N
C:	no. examples with the same value for Y and a different design for component N
D:	no. examples with the same value for Y and the same design for component N
E:	total no. examples with a different value for input feature Y
F:	total no. examples with the same value for input feature Y
G:	total no. examples with a different design for component N
H:	total no. examples with the same design for component N

- $P_{ij}$is the probability the variable is in both row i and column j
- $P_{i+}$is the marginal probability the variable is in row i
- $P_{+j}$is the marginal probability the variable is in column j.

Once a Tau value is found for each combination of input feature and solution component, we can order these combinations. In general, we expect the combinations with higher Tau values to indicate a stronger relationship between input feature and solution component (1 indicates perfect association, 0 indicates no association). For a new problem, which components can stay the same and which must be changed? Given this new problem case and the retrieved solution, we have a difference vector between the two. Where an attribute is different, we can now use these Tau values to determine which solution components must be changed in response to this difference in the input attribute.

We applied this technique to carry out adaptation in the footwear domain [12]. In practical application, it was discovered that the approach generally held true, but there were some Tau values that did not agree with an expert's mappings of input features to solution components (Section 4). Given a listing of input feature and solution component combinations ordered by Tau value, there was typically a group of the very highest Tau values that indicated correlations confirmed by domain experts. The bottom of the listing (those combinations with very low Tau values) were combinations that generally were not indicated by domain experts as correlated. There was, however, a group in the middle where some represented valid correlations, while others did not.

Another drawback of this technique is that each input feature is dealt with on an individual basis. Sometimes second-order features (the combination of several features and the compound change in them) are important in determining the related design solution. Artificial neural networks (ANNs) with a hidden layer that allows the formation of these second-order attributes could be employed.

3 Using a Neural Network to Determine What to Change

A backpropagation network could be used to determine which solution components must be changed in the retrieved case to cater for differences in the problem case. Inputs map to the difference vectors between case input descriptions while the outputs map to whether solution components should be changed or stay the same. Each output corresponds to a different case solution component.

Choosing the cases to use in a network is a problem in itself. Hanney presents the following three strategies for case selection [13]:

1. Cases with less than some threshold number of differences are compared.
2. Similar cases are compared. This requires a similarity metric for determining closeness. If a similarity metric is used in retrieval, it could be used again here.
3. Each case is compared with the n most similar cases in the case base.

A fourth strategy that is similar to the third approach above could be more effective. Taking a random sample of the case base m, retrieve the (large value) k most similar cases to the cases in m and of those $k \times m$ pairs again select a random sample. The reason we choose a second random sample rather than reducing the value of k is to allow some of the training patterns to be further apart. If only the closest cases are used to train the neural network, it could considerably cripple the learning of adaptation knowledge as this assumes that the closest cases alone will give a good coverage of the adaptation ability of the cases. As Hanney discovered, choosing only the n most similar cases learnt the least amount of adaptation rules [13].

Consider again the example in Section 1.1 of a case base with two cases whose input features' difference vector is 0 0 0.6 0 1. The neural network for this case base would therefore have five inputs and one of the training samples could have the inputs 0 0 0.6 0 1 for the difference vector above. Using a back-propagation ANN, we have supervised learning and therefore must specify expected outputs for each training example.

Outputs in this network are related to the solution components of the cases. Each separate component of a case would map to a different output neuron. The inputs are problem description difference vectors and the outputs are solution component difference vectors. So, for each training example, the expected outputs would correspond to which components could stay the same and which must be changed because of the differences in the input features. For the above example, there would be three output nodes, one for whether or not the first component needs to be changed, one for the second component, and one for the third component. For this solution difference vector where the component designs for the first case are 1A 2B and 3C, and the designs for the second case are 1A 2B and 3D, the first two components remain the same and the third one is different. Giving *same components* the value 0 and *different components* the value 1, the expected output for the inputs 0 0 0.6 0 1 would be the solution difference vector 0 0 1.

Having trained these ANNs on a set of difference vectors, retrieval consists of presenting the difference vector between the input values of the current problem and a retrieved case. The output nodes that activate (1) indicate which solution components of the retrieved case would have to be changed, and those outputs that do not activate (0) would be able to be retained from the retrieved case's solution in the current case's solution.

This technique was found to be the most effective in determining what has to be changed in case adaptation. Because of the nature of our application domain, footwear design, components of the design are either retained or changed. Within this domain, a component design cannot be changed 'a bit' as whether it changes significantly or in a minor way, the mould (or pattern) for that solution component has to be remade completely. In domains where the case solution's features can be adjusted in partial ways, the outputs from the neural network could be changed to indicate a degree of change in the solution features.

A subset of randomly selected difference vectors was adequate to train the network to achieve very good results. However, it was found that increasing the number of training examples gave better results. There would seem to be a trade-off between the number of examples used for training and the time taken to train the network.

4 Experimental Results from the Footwear Domain

This paper has described techniques for finding a difference vector similar to that between the current problem case and the retrieved case. An exact matching technique was tried and it rarely found an appropriate difference vector. A nearest neighbour technique was more successful but quite time consuming, as was a backpropagation technique to retrieve a specific difference vector. There were significant difficulties in getting these backpropagation networks to train. In most attempts, all the networks outputs converged to 0.

The automated techniques to learn generalised adaptation information from the difference vectors were considerably more successful than the attempts to find similar difference vectors. Therefore, the results of these techniques applied in the footwear domain are described in detail, after a summary of relevant terms within that domain.

- *upper:* part of the shoe that covers the top of the foot. It may consist of numerous components attached to each other. The main two parts of the upper are the vamp (covers the toes and the front half of the foot) and the quarters (covers the back part of the foot).
- *sole:* part of the shoe that covers the underneath of the foot and makes contact with the ground. It is attached and held onto the foot by the upper.
- *last:* approximately foot-shaped form over which the upper is fitted to give the shoe its shape.

Zhou and Dillon's Symmetrical Tau: On application of Zhou and Dillon's Symmetrical Tau to calculate the degree of association between any input problem feature and any solution component, we expect that the higher the Tau value generated for a particular combination, the greater the association between the input feature and the solution component. A strong association allows us to predict which components can be retained from a retrieved design and which ones must be changed. The Tau example presented here used 50 difference

Table 2. Leg length's contribution to quarters design

		Leg Length different		
		Yes	No	
Qtrs different	Yes	21	11	32
	No	0	18	18
		21	29	

vectors from randomly selected cases in the case base. For each input feature, its association with each of the four solution components was calculated, and then the combinations were ordered by Tau value. The components of the solution in a footwear design correspond to the sole, the last, the quarters and the vamp design.

To calculate a Tau value for a input feature/solution component relationship, a contingency table must be developed. For example, for the input feature leg length's association with the solution component quarters design, the number of difference vectors with a different value for leg length was calculated. The number with a different quarters design component was also calculated. Additionally the number with a different value for leg length and also with a different quarters design component was calculated, giving contingency Table 2.

The probability that a vector belongs to any given row and column P_{ij} is:
$P_{11} = 21/50$, $P_{12} = 11/50$, $P_{21} = 0/50$, $P_{22} = 18/50$

Probabilities that a vector falls into a given row P_{i+} or column P_{+j} is:
$P_{1+} = 32/50$, $P_{2+} = 18/50$, $P_{+1} = 21/50$, $P_{+2} = 29/50$

From the equation for the Tau (τ) value in Section 2, the 1st and 2nd terms of the numerator for this combination are:

$$\sum_{j=1}^{J}\sum_{i=1}^{I}\frac{P_{ij}^2}{P_{+j}} = \frac{P_{11}^2}{P_{+1}} + \frac{P_{21}^2}{P_{+1}} + \frac{P_{12}^2}{P_{+2}} + \frac{P_{22}^2}{P_{+2}} = 0.726896$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{P_{ij}^2}{P_{i+}} = \frac{P_{11}^2}{P_{+1}} + \frac{P_{12}^2}{P_{+2}} + \frac{P_{21}^2}{P_{+1}} + \frac{P_{22}^2}{P_{+2}} = 0.71125$$

The 3rd term of the numerator (also the 1st term of the denominator) is:

$$\sum_{i=1}^{I} P_{i+}^2 = P_{1+}^2 + P_{2+}^2 = 0.5392$$

The 4th term of the numerator (also the 2nd term of the denominator) is:

$$\sum_{j=1}^{J} P_{+j}^2 = P_{+1}^2 + P_{+2}^2 = 0.5128$$

Table 3. Ordered Tau Values for footwear input feature's contribution to solution components

Input feature	*Sol^n Component*	*Tau Value*	*Input feature*	*Sol^n Component*	*Tau Value*
OpenToe?	Last	5.21485e-05	DegreeofToeShape	Last	0.0885417
OpenToe?	Quarters	0.000197006	SlingBack?	Quarters	0.0915698
DegreeofToeShape	Vamp	0.00354456	SlingBack?	Vamp	0.0915698
ApronFront?	Quarters	0.00456673	DegreeofCutOut	Last	0.0969382
SlingBack?	Sole	0.00569107	HeelHeight	Vamp	0.107143
ToeSeam?	Quarters	0.00606061	LegLength	Sole	0.125418
OpenHeel?	Last	0.00620404	TypeofFastening	Sole	0.166717
OpenHeel?	Sole	0.0214646	LegLength	Last	0.168116
OpenHeel?	Vamp	0.0234375	ToeShape	Vamp	0.174626
OpenHeel?	Quarters	0.0234375	SoleThickness	Vamp	0.184244
OpenToe?	Sole	0.032882	DegreeofCutOut	Quarters	0.202812
OpenToe?	Vamp	0.0359043	TypeofFastening	Last	0.212608
ApronFront?	Last	0.0409207	HeelHeight	Quarters	0.259259
ToeSeam?	Last	0.0409207	ToeShape	Quarters	0.289406
ApronFront?	Sole	0.0447958	DegreeofCutOut	Vamp	0.289773
ToeSeam?	Sole	0.0447958	ToeShape	Sole	0.316005
LegLength	Vamp	0.0467068	HeelHeight	Last	0.336485
ApronFront?	Vamp	0.048913	ToeShape	Last	0.34593
ToeSeam?	Vamp	0.048913	Wedge?	Sole	0.369748
Wedge?	Quarters	0.0580564	LegLength	Quarters	0.407328
Wedge?	Vamp	0.0580564	SoleThickness	Last	0.46819
DegreeofCutOut	Sole	0.0613909	TypeofFastening	Quarters	0.479167
DegreeofToeShape	Quarters	0.0740741	TypeofFastening	Vamp	0.479167
SlingBack?	Last	0.0766074	SoleThickness	Quarters	0.480405
DegreeofToeShape	Sole	0.0808824	HeelHeight	Sole	0.672354
Wedge?	Last	0.0814294	SoleThickness	Sole	0.82969

Hence the Tau value for the association between leg length and quarters is:

$$\frac{0.726896 + 0.71125 - 0.5392 - 0.5128}{2 - 0.5392 - 0.5128} = 0.407327$$

Comparing this with the Tau for leg length's contribution to the sole component (0.125418), leg length has a significantly greater contribution to the quarters design than to the sole design, as expected. From the ordered Tau values in Table 3 for all the attributes' contributions to the different components, the values over 0.5 show the strongest associations and are the most obvious; that is sole thickness and heel height's contribution to sole design. Values between 0.3 and 0.5 are still quite strong associations while values under 0.3 are rather hit and miss. Here, associations known to be strong in the domain are interspersed with weak associations, perhaps influenced by the number of examples (or lack thereof) displaying the difference in the attribute being evaluated. For example, the smallest value of Tau; namely, open toe's contribution to last design, is very small whereas it should be quite a strong association. The number of difference vectors showing a difference in open toe was only three and by chance, in those three examples, the association is not demonstrated. In general, the higher values are better indicators, but there are still hit and miss values up to 0.3. It may be possible to increase the accuracy by increasing the number of difference vectors used to calculate the Tau values. Increasing the number of examples, however, will not change the lack of second-order features and their influence on the various solution components.

In the footwear domain, there are two values, sole thickness and heel height, that are known by domain experts to describe the main requirements of the sole. While the Tau values show that if either or both of these values change then the sole design must change, these values may also influence last design. If one or the other of these values is different then the last must also be changed. If both change the same amount then the same last may be able to be used as there is an implicit second-order attribute of pitch (that is the difference between the heel height and sole thickness). Without this expert knowledge input, the Tau will not take into account pitch and any other second-order attributes.

ANNs for learning adaptation knowledge: Using a backpropagation network for case adaptation was the most successful approach. Inputs were the values that made up the difference vector between two shoe designs' input features, while the outputs were related to each component of the shoe design.

Several different-sized networks were tried, and it was found that a mid range network was the most effective (a network with 15 hidden layer nodes performed better than networks with 10 or 20 hidden layer nodes). The networks generally trained quite quickly, indicating that the relationships between the inputs and outputs were straightforward to determine. The outputs that were not correct showed associations that were generally not present in the training data.

Another newly constructed test set was then tested on a network trained on the original difference vectors. The set was also tested on another ANN with the same configuration but trained on the original difference vectors combined with the original test data. Increasing the number of training vectors from 60 to 75 allowed the network to learn several more associations, as there were several vectors in the first test set that showed associations that did not exist in the original training set. This addition of training vectors increased accuracy from 91.25% to 96.25%.

4.1 Conclusion

Difficulties were found with exact matching, nearest neighbour matching and ANN matching of the difference vector between the problem case and a retrieved case, with stored difference vectors in the case base.

The chance that the difference vector between a current problem and a retrieved case would exactly match an existing difference vector is small. Using the existing difference vectors (or a subset of them) to acquire generalised adaptation knowledge was significantly more successful.

Using these difference vectors with a data mining technique such as Zhou and Dillon's Symmetrical Tau, it was possible to prioritise the associations between input features and solution components to some degree. There were limitations in that strong real-life associations identified by a domain expert were often prioritised lower if there were not many examples in the training set, and also the approach was unable to find and prioritise second-order features.

A backpropagation network for learning adaptation knowledge, where each output mapped to a design component for reuse or change, was the most successful technique. We found that this approach resulted in a high frequency of correct recommendations for solution components to change or reuse, as verified by a domain expert. There were no apparent drawbacks to this approach and it would seem to be a widely applicable and effective automated technique for case adaptation tasks.

References

1. Main, J., Dillon, T., Shiu, S.: A tutorial on case based reasoning. In: Soft Computing in Case Based Reasoning, pp. 1–28. Springer, Heidelberg (2001)
2. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
3. Kolodner, J.: An introduction to case-based reasoning. Artificial Intelligence Review 6(1), 3–34 (1992)
4. Leake, D.: Becoming an expert case-based reasoner: Learning to adapt prior cases. In: Proceedings of the Eighth Annual Florida Artificial Intelligence Research Symposium, pp. 112–116 (1995)
5. Anand, S., Patterson, D., Hughes, J., Bell, D.: Discovering case knowledge using data mining. In: Wu, X., Kotagiri, R., Korb, K.B. (eds.) PAKDD 1998. LNCS, vol. 1394, pp. 25–35. Springer, Heidelberg (1998)
6. Corchado, J., Lees, B., Fyfe, C., Rees, N., Aiken, J.: Neuro-adaptation method for a case-based reasoning system. Computing and Information Systems 5(1), 15–20 (1998)
7. Lofty, E., Mohamed, A.: Applying neural networks in case-based reasoning adaptation for cost assessment of steel buildings. International Journal of Computers & Applications 24, 28–38 (2003)
8. Salem, A.B., Nagaty, K., Bagoury, B.E.: A hybrid case-based adaptation model for thyroid cancer diagnosis. In: ICEIS 2003, vol. 2, pp. 58–65 (2003)
9. Swingler, K.: Applying Neural Networks: A Practical Guide. Academic Press Inc., London (1996)
10. Zhou, X., Dillon, T.: A statistical-heuristic feature selection criterion for decision tree induction. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(8), 834–841 (1991)
11. Sestito, S., Dillon, T.: Automated Knowledge Acquisition. Prentice Hall, Englewood Cliffs (1994)
12. Main, J., Dillon, T.: A hybrid case-based reasoner for footwear design. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 499–509. Springer, Heidelberg (1999)
13. Hanney, K., Keane, M.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 359–370. Springer, Heidelberg (1997)