Case-Based Reasoning

In Case-Based Reasoning (CBR), solutions to previously-solved problems are reused to solve new problems. We have looked at CBR before in this course: as a way of speeding-up search on iconic representations. (In fact, McGinty's & Smyth's work was not only a way of speeding up search, but also a way of 'personalising' search because they used a different case base for each user.)

We're now going to look at CBR a little more systematically.

1 CBR: A Recap

CBR is a theory about how humans solve problems:

- *Reasoning is remembered.* We store our problem-solving experiences (both successful and unsuccessful).
- *Remembering is reasoning.* When we are faced with a new problem, we *recall* similar previously-solved problems, to avoid solving the new problem 'from scratch'.

It is the basis of many successful agents. It is suitable in domains where:

- Similar problems have similar solutions.
- Problems recur.

It can be a 'total' solution: CBR says something about knowledge acquisition, problem-solving and learning.

A CBR system has a *case base* containing a number of *cases*. Cases are quite different from rules. Rules are generalised knowledge. They express in a single statement regularities about the world. Cases, on the other hand, provide knowledge about problem-solving episodes: they describe specific instances, rather than abstracting over many instances.

Each case typically comprises at least two parts: a description of the problem, and a description of its solution. One of the things that makes case acquisition relatively easy is that you do not have to understand *how* problems were solved. You only have to record that such a problem was solved in such a way.

Suppose you are now faced with a new problem. This new problem is sometimes called the *probe*. The case base is searched for a case that describes a similar problem. The most similar case is retrieved. (Sometimes more than one case is retrieved but we'll ignore this to simplify the explanation.) Its solution is then adapted so that it is applicable to the probe problem.

The probe with its (adapted) solution can later be added to the case base. This augments the base of experience that the system can use to tackle subsequent problems. It is therefore a very easy form of machine learning.

We'll now look more closely at different ways of representing cases, doing retrieval and doing adaptation.

2 The Cases

In CBR, cases are most commonly represented by a set of attribute-value pairs. Some of the attribute-value pairs will describe the problem; and one or more will describe the solution to the problem. Here's an (incomplete) example of a case. It comes from a system that advises on the setup of aluminium pressure die-casting machines.

Problem Description	
weight_of_casting	240.00
projected_area_of_casting	19.50
average_number_of_impressions	1.00
machine_type	T400
metal_type	LM24
Solution	
cavity_fill_time	13.77
plunger_velocity	225.00
pressure_on_metal	8000.00

A probe case would, obviously, consist of just the problem description.

Although attribute-value pairs are common, there are other possibilities. In some CBR systems, the parts of the case are represented simply by strings of text; in other systems the problem part of the case is represented by question-answer pairs. In other systems, we resort to using richer data structures: trees, directed acyclic graphs (DAGs) and even general graphs. Our presentation is going to assume that we have a set of attribute-value pairs as this is the simplest to explain and the commercially most prevalent option. (But note that the use of XML, which gives tree-like structures, is starting to rival the attribute-value pair approach.)

Here is another example using an attribute-value pair representation. Consider a user who is trying to decide what meal to prepare for dinner guests. The probe is on the left and there are two cases on the right. We will be using this example in subsequent sections of these notes.

calories	0
cuisine	Mexican
containsNuts	false
spiciness	killer
meatType	turkey

calories	700
cuisine	Greek
containsNuts	false
spiciness	mild
meatType	lamb
recipe	Nico's Lamb Casserole

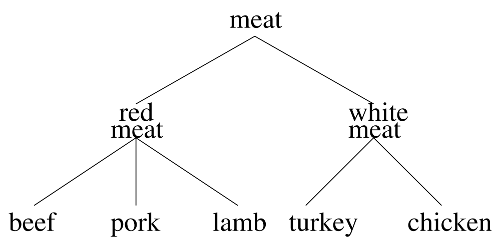
calories	650
cuisine	Indian
containsNuts	yes
spiciness	hot
meatType	chicken
recipe	Nutty Chicken Vindaloo

Each attribute has a data type associated with it. The obvious data types are allowed, e.g. boolean, numeric and strings. But CBR systems usually allow a much wider range of user-defined data types as well. Here are two examples:

- They might allow an ordered enumerated type. E.g. the **spiciness** attribute might have as its type the following:

  mild $<$ medium $<$ hot $<$ extra_hot $<$ killer

- They might allow an enumerated type on which a taxonomy (class hierarchy) has been defined. E.g. the **meatType** attribute might have the leaves of this taxonomy as its type:



3 Retrieval

In CBR, to compute the similarity of cases in the case base to a probe case requires the definition of a similarity function. Usually, a *global similarity function* is built up from a number of so-called *local similarity functions*, one per attribute. Most often, the global similarity is a weighted sum of the local similarities. The weights allow different attributes to have different importances in the computation of the overall similarity. Weights would typically lie anywhere between zero and one; a weight of zero would indicate a totally irrelevant attribute.

The weights can be supplied by the knowledge engineer and/or the domain expert or, in some cases, they may even be user-supplied. There are also ways of learning the weights from training data (much as per neural nets).

Let p be the probe, let c be a case, let there be n attributes in the problem description part of cases, let p_i and c_i be the value of the ith attribute in p and c, let sim_i be the local similarity function for attribute i and let w_i be the weight of the ith attribute, then the global similarity of p to c, $\mathrm{sim}(p, c)$ is defined as follows:

$$\mathrm{sim}(p, c) \triangleq \frac{\sum_{i=1}^{n} w_i \times \mathrm{sim}_i(p_i, c_i)}{\sum_{i=1}^{n} w_i}$$

Having computed the similarity of the probe to all the cases in the case base, the system retrieves those with the highest similarity scores. In some systems, those with similarity scores that exceed a threshold are retrieved. In other systems, the top n (e.g. the top 10 or 20) are retrieved.

For our example, here are possible local similarity functions:

calories: For numeric types, the obvious approach is to start with a distance function:

$$\mathrm{dist}(x, y) \triangleq \mathrm{abs}(x - y)$$

But then we must normalise to get a value in the range $[0, 1]$. You do this by dividing by the maximum distance you think is possible, e.g. for calories, the minimum is 0 and the maximum for a meal is, say, 1500, so the maximum distance is 1500 - 0 = 1500. Then to turn a normalised distance function into a similarity measure, you subtract from 1:

$$\mathrm{sim}(x, y) \triangleq 1 - \frac{\mathrm{dist}(x, y)}{\mathrm{maxRange}}$$

cuisine: For symbolic types, the simplest similarity measure is to score 1 for equality and 0 for inequality:

$$\mathrm{sim}(x, y) \triangleq \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

(Cuisine could have a better similarity measure if we were to use more knowledge, e.g. along the lines of what we do with meatType below.)

containsNuts: Boolean types would use the same similarity measure as we used for cuisine.

spiciness: We could use the same similarity measure yet again, but if we did, we'd be ignoring the ordering that we defined on these values. A better approach is to use a normalised distance function (and subtract from 1). The distance function will be based on how far away in the ordering one item is from another, e.g. medium and killer are 3 away from each other. The range we use for normalisation will, in this case, be 4.

meatType: Again we use a normalised distance function but this time it uses distance in the taxonomy. For example, the distance between chicken and turkey is 2, but the distance between chicken and lamb is 4. The range for normalisation is 4.

Question. *Let's use the probe and the two cases we gave earlier. Suppose the weights are calories 0.2, cuisine 0.6, containsNuts 0.9, spiciness 0.6 and meatType 0.5.*

What are the similarities?

4 Adaptation

In many CBR systems, especially those in commercial use, adaptation is left to the user. It is not done automatically. However, we will look at the possibility of automatic adaptation in this section.

The solution can be adapted by the application of formulae or rules:

Parameter adjustment: If the probe and the retrieved case differ, then it may be that some value in the solution needs to be adjusted proportionately. For example, if the probe concerns the casting of a 250 kilogram object but the retrieved case describes the casting of a 240 kilogram object, then values in the solution may need to be adjusted (perhaps the pressure needs to go up by a proportionate amount). This might be done by a condition-action rule:

if probe's weight - case's weight > 5 **then** solution pressure := solution pressure * 1.2

Substitution: If the retrieved solution is not compatible with the probe (some constraint is violated), then it might be possible to make it compatible by replacing a value in the solution by another value.

if solution pressure > 6000 and probe's metal type = LM24 **then** plunger velocity := 60

Of course, the problem with all this is that it increases the amount of work involved in building the system: the knowledge engineer must work out all these adaptation rules. (Some progress has been made in learning some of this knowledge.)

5 How Deep is an Agent's Model of the World?

Increasingly deep models

Increasing knowledge emgineering

Model-Based Reasoning	Rule-Based Reasoning (using Heuristic Associations)	Case-Based Reasoning	Information Retrieval

Obviously, this is only suggestive of the relationship between different approaches to building agents.

Information Retrieval: You can use an IR system to help you to solve problems. But the IR system has no real model of the domain. It is doing little more than matching user keywords with keywords in an arbitrary collection of documents. In some cases, use might also be made of a thesaurus or of statistical properties of the documents (e.g. word frequencies).

CBR: As we've seen, CBR has quite shallow knowledge; it can even be used in domains where there is little or no understanding of the domain.

Rule-Based Reasoning: As we've discussed, for the most part, rule-based expert systems try to capture heuristic associations used by human experts.

Model-Based Reasoning: In model-based reasoning, the knowledge base contains a model of the system being reasoned about. The model captures the structure and/or behaviour of the physical system. It uses *deep knowledge* from science and engineering, rather than an expert's heuristic associations.

(The CASNET system that we looked at in the previous lecture sits somewhere between Rule-Based Reasoning and Model-Based Reasoning.)

Since it may not be clear, let's try to tease out the difference between model-based reasoning and rule-based reasoning using heuristic associations.

The first point is that both might use rules. But, if there are rules in a model-based reasoning system, then they tend to be causal rules, as opposed to the diagnostic rules used in rule-based expert systems.

But the real contrast is that, for rule-based expert systems, we require experts from whom we can elicit heuristic associations experience; in model-based reasoning, we require a good enough understanding that we can capture the structure and function of the domain in a reasonably complete, consistent and correct way. The model-based reasoning approach has 'deeper' knowledge of the domain.

Model-based reasoning requires such a good knowledge of the system that it is feasible only for simple, well-understood domains. The functioning (and malfunctioning) of physical devices, such as electrical circuits, is a good example. Model-based reasoning could not be used for most medical domains, since our knowledge of the structure and functioning of the human body is at once both too complex and yet too incomplete. In these domains, rule-based reasoning using heuristic associations or CBR can be used.

Here's another picture that is suggestive of the up-front effort and the on-going effort needed by the different kinds of systems we have looked at.