Case-Based Reasoning

Bryan Blackford
- A* Cost function
- Homework
- Final
- Evals
- Projects
Solving problems by remembering similar problems
Adapting old solutions to new problems

Desired story: A knight does something that results in his death.
Abstraction: A knight does something that results in a death.

Old story: “A knight named Lacelot fought a troll with his sword and killed it.”
Adapted story: “A knight killed himself by fighting himself with his sword.”
Using a database of past experiences to create new experiences that solve new problems.

[Chris Riesbeck at Northwestern]
What features to store?

- Useful for prediction
- Handles new data
- Automatic indexing
Data structure for features

- Efficient storage for retrieval
- Feature Hashing
Search case base for similar problems

- Nearest neighbor

\[
similarity(Case_1, Case_2) = \frac{\sum_{i=1}^{n} w_i \times \text{sim}(f_i^L, f_i^R)}{\sum_{i=1}^{n} w_i}
\]

[L. Karl Branting]
Search case base for similar problems

- Inductive Retrieval

[. Karl Branting]

[L. Karl Branting]
Modifying the old solution for the new problem

Insert, Delete, or Substitute values

- Domain specific
Applying the new solution

Evaluate results

- Failure improves next attempt
- Reuse successful strategy
New solutions are added to the case base
Generating new solutions

Rules pre-process experience
Adapts solutions and problems
TRAM – Transform Recall Adapt Method

Transform problem into a related problem
Retrieve a problem-solution
Adapt solution to the original problem

TRAM: Cross-Domain-Solution

- Transform strategy: find a new problem domain with similar actions and actors, and analogically map current problem to new domain
- Adapt strategy: Map any discovered solutions back to the original domain by reversing analogical mapping

Original Problem: create a scene in which a knight accidentally meets a princess
Transformed Problem: create a scene in which a businessman accidentally meets somebody

TRAM: Generalize-Constraint

- Transform: select and generalize a feature (call it $generalized-feature) of the scene specification. Use this new scene specification as an index for imaginative recall.
- Adapt: adapt the recalled solution to the current problem by adding $generalized-feature back to the recalled scene
Problem solving with TRAMs

- Transform Original Problem
- Imaginative Memory
- Adapt Past Solutions
- Assess Solutions

Active TRAM

TRAM: Standard-Problem-Solving & Other TRAMs

Episodic Memory

Domain Assessments Boredom Assessment

Problem Specification → Solution
Imaginative Memory: Recursive TRAM application

Figure 2.6 Recursive Creativity
Applying recursive TRAMs

TRAM:Similar-Outcomes-Partial-Change
Transform: Replace the resulting state caused by an action with a related state.
Adapt: Replace state change in recalled episode with original state.

- Apply TRAM:SOPC to “A knight purposefully kills himself” to create “A knight purposefully injures himself”
- Recursively apply TRAM:GC to “A knight purposefully injures himself” to create “Someone purposefully injures themselves”
- Recall: “A lady of the court made herself ill by drinking poison”
- Adapt (using TRAM:GC) to “A knight made himself ill by drinking poison”
- Adapt (using TRAM:SOPC) to “A knight killed himself by drinking poison”
Author-Level Planning

**Figure 3.1 Author-Level Processes**
Further Reading

Scott R Turner’s dissertation
Minstrel Remixed - http://minstrel.soe.ucsc.edu/
Turner’s Minstrel - http://grandtextauto.org/2006/03/01/turners-minstrel-part-1/

*An Introduction to CBR –
Roger Schank’s development of CBR –
http://www.rogerschank.com/biography.html

Rapid Retrieval Algorithms for CBR -
Extending Case-Based Planning with Behavior Trees –
http://gaia.fdi.ucm.es/aigaion2/index.php/attachments/single/83
Questions?
Thanks

Check out CMPS 148