1. Introduction to CBR
2. Knowledge and Case Representation
3. Similarity
4. Similarity-Based Retrieval
5. Solution Adaptation
6. Learning in Case-Based Reasoning
7. Applications
8. References

Case-Based Reasoning is ...

- an approach to model the way humans think
- an approach to build intelligent systems

Central Ideas:
- store experiences made ➔ as cases
- solving a new problem do the following
  - recall similar experiences (made in the past) from memory
  - reuse that experience in the context of the new situation (reuse it partially, completely or modified)
  - new experience obtained this way is stored to memory again

Classification of CBR (I)

- sub-discipline of Artificial Intelligence
- belongs to Machine Learning methods
  - learning process is based on analogy ➔ not on deduction or induction
  - best classified as supervised learning
  - in the domain considered, the model learnt can then be used, for example, for prediction or classification purposes
  - most „work“ (calculations) are done when building the model
  - by contrast: in a memory-based approach, most calculations are done at the time of application, i.e. when doing prediction or classification
  - „most, but not all“: the necessary calculations at application time can be supported and prepared by creation of suitable data structures (e.g. kd-trees) at storing time
  - therefore, memory-based approaches are sometimes also called „lazy learning“
Problem Solving by Case

Battery Voltage

Store observations define a new problem

11.05.2010: 12.6V

Note: The new problem is a "case" without solution part.

State of Light Switch

A description of the symptoms

Case: 1998

find a case similar to the current problem and reuse the repair strategy

13.6V

CASE 1

store a collection of cases in the case base

State of Light

description of a repair strategy

Year

When a domain theory does not exist, but example cases are easy to find.

State of Light Switch: front light fuse defect

When an expert in the domain is not available, is too expensive, or is incapable of articulating verbally his performance, but example cases are easy to find.

Battery Voltage: bulb defect

When it is difficult to specify domain rules, but example cases are easy to find.

Repair

Problem with similar solutions have similar problem descriptions.

A case describes a particular diagnostic situation.

– i.e. there exists a similarity metric for problem descriptions and a corresponding set of adaptation rules

– When cases with similar solutions have similar problem descriptions.

A case is not a general rule.

When a case base already exists.

State of Lights

description of the failure and the cause

Car

Problem

When is CBR of Relevance?

Problem (Symptoms): front light does not work

A New Problem (Query) Has to Be Solved

Case Base with Two Cases

– we make several observations in the current situation

– Problem (Symptoms): front light does not work

– observations define a new problem

– when no known problem is the most similar one

– all cases are independent of one another

– not all attribute values have to be known

– Note: The new problem is a "case" without solution part

Question:

Case (Symptoms):

– How to rank the cases according to their similarity?

– How to reuse the solution of the corresponding case?

Note:

– Similarity is the most important concept in CBR.

– Similarity may be assessed based on the similarity of each feature, while the importance of different features may vary (feature weighting).
A Simple Example Scenario:
Reuse and Retain (IV)

- **Reuse**
  - adapt the solution
  - how do differences in the problem affect the solution

- **Retain**
  - if diagnosis is correct: store new case
  - add case to case base

Advantages of CBR (I)

- **Avoidance of High Knowledge Acquisition Effort**
  - case knowledge is usually easily acquirable
  - not much general knowledge required

- **Simpler Maintenance of the Knowledge in the System**
  - maintenance by adding/removing cases from the case base
  - cases are independent of one another and easily interpretable (even for non-experts)

- **Facilitation of Intelligent Retrieval** (compared to data-base systems)
  - DBMS often give too few/many results

Typical Application Fields (I)

- **Analytical Tasks**
  - Classification
  - Prediction

- **Synthetic Tasks**
  - Configuration
  - Planning

Remarks concerning analytical tasks:
- main focus is on analysing a given situation
- classification (assign objects to a class \([K_1,...,K_n]\))
  - e.g. recognition of sponges
- diagnosis (classification + verification + therapy)
  - e.g. fault diagnosis in Airbus engines
- evaluation/regression (like classification, but assignment of real-valued assessments)
  - e.g. credit risk assessment
- decision support (search for specific information relevant for decision-making)
  - e.g. web-based product catalogues, like online travel agencies
- prediction (like classification + time dependency)
  - e.g. prediction of the probability of failure of a machine’s part

Advantages of CBR (II)

- **High Quality of Solutions for Poorly Understood Domains**
  - case-based systems can be made to retain only “good” experience in memory
  - if only little adaptation is necessary for reuse, this will not impair the solution’s quality too much

- **High User Acceptance**
  - provided solution corresponds to actual experience
  - may increase trust in the solution
  - selected case and solution adaptation can be explicitly presented to the user
  - problems of rule-based / neural network-based systems
    - black boxes
    - inference process is not traceable or hidden
  - provided solutions are difficult to explain

Typical Application Fields (II)

- **Remarks concerning synthetic tasks:**
  - main focus is on composing a complex solution from simpler components
  - focus is often on solution adaptation
  - configuration: e-commerce scenario → product configuration (e.g. personal computers)
  - design: reuse of construction plans in civil engineering
  - planning: production planning

CBR Cycle (R4, [Aamodt & Plaza, 1994])

- retrieve: find most similar case(s)
  - similarity measures
  - explanation-based methods
  - case-base organisation (data structures)

- reuse: transform/adapt solution
  - different types of solution transformation (none, interactive, derivational, etc.)
  - different methods (rule-based, constraint satisfaction, model-based etc.)

- revise: verify/improve solution
  - no verification
  - verification by simulation
  - verification in the real world

- retain: keep the experience made
  - learn new cases
  - learn similarity assessment
  - learn case base organization
  - learn solution adaptation

Remarks concerning synthetic tasks:
CBR for Classification (I)

- A classifier for a set $M$ is a mapping $f: M \rightarrow I$ (where $I$ is a finite index set).
  - A case-based classifier is given by a case base, a similarity measure and the principle of the nearest neighbour.
- Definition: Given a case base $CB$, a similarity measure $sim$ and an object (problem) $q \in M$, we call $c=(p,s)\in CB$ the Nearest Neighbour to $q$, if: for all $(p',s')\in CB$ it holds $sim(q,p) \geq sim(q,p')$.
- Definition: In Nearest-Neighbour Classification each new object (query) $q \in M$ is assigned the class $s \in I$ of $q$'s nearest neighbour in $CB$, i.e. when
  $$NN = (p_{NN}, s_{NN}) = \arg \max_{c \in CB} sim(q,c)$$
  then $q$ is assumed to belong to class $s_{NN}$.

CBR for Classification (II)

- Note: The classifier defined by the pair $(CB, sim)$ is not unique, if there is more than one nearest neighbour.
- Extension to $k$-NN Classification:
  - The $k$ most similar neighbours of $q$ are considered. Typically, a majority voting is applied to determine the class of the query $q$.
  - Formally: Let $NN_k(q)=((p_1,s_1),(p_2,s_2),\ldots,(p_k,s_k))$ denote the set of $k$ nearest neighbours of $q$. If we denote by $n_i = \sum_{j} I(s_j = i)$ the frequency of class label $i$ within the $k$ nearest neighbor, then $q$ is assumed to belong to class $s = \arg \max_{i} n_i$.

2. KNOWLEDGE AND CASE REPRESENTATION

What forms of knowledge are parts of a CBR system?
How can cases be represented?

Knowledge Container Model [Richter, 1989]

- „In order to solve problems, one needs knowledge."
- Knowledge of a CBR System
  - vocabulary: knowledge representation
  - retrieval: similarity assessment (measures)
  - solution transformation: rules
  - cases
- Knowledge Management
  - as the environment may change, maintenance of the containers' contents over the lifetime of the CBR system is crucial to guarantee its continued usability

Case Contents

- Problem / Situation Information
  - must cover all the information that is necessary to decide if this case is applicable for a new situation
  - target of the problem
  - constraints
  - characteristics
  - new situation = query

Solution
- contains all the information that describes a solution to the problem sufficiently accurately
  - solution itself
  - justifications
  - possible alternative solutions
  - steps that were tried, but failed

Solution Evaluation
- feedback from the real world
  - How good was the solution for the problem?
**Case Representation Formalisms (I)**

**Attribute-Value Based Case Representation**
- Case (problem and solution) is represented by pairs of attributes and belonging values.
  - e.g.: price = 9.95€
- Set of attributes \( A = \{A_1, \ldots, A_n\} \) is (in general) fixed for all cases.
- To each attribute \( A_i \) there is an associated domain \( D_i \) and for each attribute's value \( a_{iD} \), e.g.
  - numerical attributes (integer or Real or subsets of those)
  - symbolic attributes (finite domains, \( D_i = \{d_{i1}, \ldots, d_{in}\} \))
  - textual attributes (strings)
- **Note:**
  - Choice of attributes and corresponding domains to represent cases represents general knowledge: vocabulary knowledge.
  - Choice of domains is mainly influenced by the requirements for similarity computation and solution adaptation.

**Case Representation Formalisms (II)**

- **Choice of attributes**
  - must allow for the decision whether a case and a new situation are similar
  - should avoid redundancies
  - should represent independent properties of a case
- **Disadvantages**
  - no structural or relational information is representable
  - no ordering information (e.g. sequence of actions) is representable
- **Advantages**
  - straightforward representation
  - easy to understand and implement
  - cases are easy to store (usage of databases)
  - efficient retrieval
- **Example**
  - recall the example from the Introduction

**Case Representation Formalisms (III)**

**Object-Oriented Case Representation**
- Refinement and more structured extension of attribute-value based representation
- Compositing of related attributes to object descriptions; each object is described by a fixed set of attributes
- Case = Set of Objects

**Graph- and Tree-Based Representation**
- e.g. suited for atomic/molecule structures or electrical circuit designs

**First-Order Based Case Representation**
- problems and solutions are represented as sets of Grundatome (variable-free)

**Hierarchical Case Representation**
- each case is represented on several levels of abstraction
- Generalised Cases
- each case describes sets of cases at once, which are highly similar to one another
- smaller case bases, simplified case/solution adaptation

**3. SIMILARITY**

When is a new problem (query) similar to a case’s problem part?

What forms of similarity measures are suitable?

**Meaning of Similarity**
- Similarity is the central notion in Case-Based Reasoning.
- Similarity is always considered between problems (not solutions of cases).
- Selection of cases during the “Retrieve” phase is based on the similarity of cases to a given query.

**Observation I:** There is no universal similarity; similarity always relates to a certain purpose.
- e.g. two cars can be similar if they have the same max speed or cost approximately the same \( \rightarrow \) different aspects of similarity.

**Observation II:** Similarity is not necessarily transitive.
- e.g. 10€ are similar to 12€, 12€ are similar to 14€, \( \ldots \), 10€ are similar to 100€. But: 10€ are not similar to 102€ \( \rightarrow \) property of “small numeric difference” is intransitive

**Observation III:** Similarity does not have to be symmetric.

**Similarity and Utility**
- Purpose of Similarity: Selection of solutions that can be easily transferred / adapted to the problem at hand.
- Similarity = Utility for Solving a (new) Problem
- **Note:**
  - Utility is an a-posteriori criterion: In general, the utility (of a case) can be estimated after having solved the problem.
  - Similarity concerning problem situations is an a-priori criterion: Similarity must be estimated before solving the problem.
- **Goal:** Similarity must approximate utility as accurately as possible.
Similarity Measures

- **Idea:** Numerical modelling of similarity, capturing the degree of similarity
- **Definition:** A *Similarity Measure* on a set $M$ is a real-valued function $\text{sim}: M^2 \to [0,1]$. We say that $\text{sim}$ is
  - reflexive iff. $\forall x \in M: \text{sim}(x,x) = 1$
  - symmetric iff. $\forall x,y \in M: \text{sim}(x,y) = \text{sim}(y,x)$
- Beyond ordinal information, similarity measures allow for a quantitative statement on the degree of similarity.
- **Definition:** Each similarity measure induces a *similarity relation* $R_{\text{sim}}$ as $R_{\text{sim}}(x,y)$ iff. $\text{sim}(x,y) \geq \text{sim}(u,v)$.

Distance Measures

- **Definition:** A *Distance Measure* on a set $M$ is a real-valued function $d: M^2 \to [0,\infty)$. We say that $d$ is
  - reflexive iff. $\forall x \in M: d(x,x) = 0$
  - symmetric iff. $\forall x,y \in M: d(x,y) = d(y,x)$
- **Definition:** Each distance measure induces a *distance relation* $R_d$ as $R_d(x,y)$ iff. $d(x,y) \leq d(u,v)$.

Relation Between Distance and Similarity Measures

- **Definition:** A similarity measure $\text{sim}$ and a distance measure $d$ are called *Compatible* if and only if
  $\forall x,y,u,v \in M: R_{\text{sim}}(x,y,u,v) \iff R_d(x,y,u,v)$
- **Lemma (Measure Transformation):** If there is a bijective, order-reversing mapping $f: [0,1] \to [0,1]$ with $f(0) = 1$ and $f(d(x,y)) = \text{sim}(x,y)$ then $\text{sim}$ and $d$ are compatible.
- **Note:** A transformation function $f$ can be employed to construct a compatible pendant for a given $\text{sim}$ or $d$, respectively.
- **Examples:**
  - $f(x) = 1 - x$ (for attribute values $x$ with $0 \leq x \leq 1$)
  - $f(x) = 1 - x/x_{\max}$

Examplary Similarity Measures (I)

*for attribute-value based case representations*

- **Hamming Distance**
  $H(x,y) = n \cdot \frac{\sum_{i=1}^{n} |x_i - y_i|}{n}$
  - for binary features $\forall x \in \{0,1\}^n$:
    $H(x,x) = 0$, $H(x,y) = H(y,x)$
  - $H(x,y)$ is the number of attributes with differing values
  - $H$ is a *distance measure*:
    $H(x,y) = H(y,x)$
  - $H$ is a *distance measure*:
    $H(x,y) = H(y,x)$

- **Simple Matching Coefficient (SMC)**
  $\text{SMC}(x,y) = \frac{a + b - c}{a + b + c + d}$
  - $a = \sum x_i y_i$, $b = \sum (1-x_i) y_i$, $c = \sum x_i (1-y_i)$, $d = \sum (1-x_i) (1-y_i)$
  - transformation of the Hamming distance into a compatible similarity measure by $f(0) - 1 - d_{\text{ham}}$ yields the simple matching coefficient

Examplary Similarity Measures (II)

*for attribute-value based case representations*

- **Measures for Real-Valued Attributes**
  - $x,y \in \mathbb{R}$ for all $i$
  - generalisations of the Hamming distance
    - city block metric $d_1$
      $d_1(x,y) = \sum_{i=1}^{n} |x_i - y_i|$
    - Euclidean distance $d_2$
      $d_2(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
    - weighted Euclidean distance $d_w$
      - $d_w(x,y) = \sum_{i=1}^{n} w_i(x_i - y_i)^2$
      - weights $w_i$ must be specified
    - $p$-norm $d_p$
      $d_p(x,y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{1/p}$

Note: Similarity measures for other case representations (e.g., object-oriented, graph-based, etc.) are not considered in this lecture; see literature references.
Problem Solving by Case

**Exemplary Similarity Measures (III)**

**Measures for Sparsely Filled Cases**
- In some domains, the value "0" is dominating which should be taken into consideration by a distance / similarity measure.
- Example:
  - A case describes a customer. Each attribute describes how many times the customer has bought a specific product. There are 1000 different products, hence a case comprises 1000 attributes.
  - Customer A and B have bought one product each, but different ones.
  - Their Euclidean distance is \( \sqrt{2} \).
  - Customer C and D have bought 100 different products each. 95 of them are identical.
  - Their Euclidean distance is \(< 10\).
- Thus, A and B are more similar than C and D.

**Exemplary Similarity Measures (IV)**

**Measures for Sparsely Filled Cases**
- If zeros are dominating, a generalization of the SMC is applied.
- Let \( n \) denote the number of attributes and \( f \) the number of attributes, in which both cases are equally zero. Then, we define

\[
SMC_n(x,y) = \frac{\sum_{i=1}^{n} (s_{i,x} - s_{i,y})}{n - f}
\]

- In the example from the previous slide:
  - \( SMC_n(A,B) = (998-998) / (1000-998) = 0 \)
  - \( SMC_n(C,D) = (990-895) / (1000-895) = 0.905 \)

**Local-Global Principle**
- case description by \( n \) attributes \( A_1, \ldots, A_n \)
- each attribute has a certain type \( T_i \) (e.g. numeric)

**Local Similarity**
- a separate similarity function is used for each attribute:
  - \( \text{sim}_{T_i}(x_i, y_i) \rightarrow [0,1] \)
  - local measures are depending on the respective type \( T_i \) of the attribute \( A_i \)

**Global Similarity**
- \( \text{sim}(x,y) = \sum_{i=1}^{n} \text{sim}_{T_i}(x_i, y_i) \)
- \( F(0.1) \rightarrow [0,1] \) **Amalgamation Function**
- requirements on \( F \):
  - \( F \) is monotonous in each of its arguments
  - \( F(0,\ldots,0) = 0 \) and \( F(1,\ldots,1) = 1 \)

**Local Similarity Measures (I)**

**Similarity Tables**
- for attributes with symbolic type \( T_{x} = \{ s1, \ldots, s_m \} \)
- sim table/matrix \( \text{sim}(x,y) = \{ s(x,y) \} \)

<table>
<thead>
<tr>
<th>Type</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>1</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>DDR</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>OR</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Difference-Based Similarity Functions**
- for attributes with numeric type (e.g. integer or real-valued)
- similarity is based on the numerical difference between case and query value
  - \( f \): \( x \rightarrow (x - \text{mean}(x)) \)
  - \( g \): \( y \rightarrow (y - \text{mean}(y)) \)
- typical requirements on \( f \):
  - \( f(x,y) = f(y,x) \)
  - \( f(x,y) \leq f(x,a) + f(a,y) \)
  - \( f(x,y) = 0 \) if \( x = y \)

**Local Similarity Measures (II)**

**Other Types**
- ordered symbolic data types
  - e.g. \( T_{x} = \{ \text{small average}, \text{large average} \} \)
- taxonomic data types
  - elements of \( T_i \) can be arranged within a taxonomical (tree) structure
  - e.g. attribute to describe the types of CPUs

**Examples:**
- weighted average
  - \( F(\ldots,s_n) = \sum_{i=1}^{n} w_i s_i \)
  - \( \text{Maximum} \)
  - \( F(\ldots,s_n) = s_n \) with \( s_{i=1} = \ldots = s_{n-1} \)
  - \( \text{Minimum} \)
  - \( F(\ldots,s_n) = s_1 \) with \( s_{i=2} = \ldots = s_{n} \)
- background knowledge included (for \( q \neq c \))

- **not considered here**
4. SIMILARITY-BASED RETRIEVAL

How to retrieve a query’s nearest neighbour(s)?

Two-Stage Retrieval

- Idea: MAC/FAC (many are called, few are chosen)
  1. preselection of possible solution candidates \( M_k \subset CB \)
     where \( M_k = \{ c_i \in CB \mid fac(c_i) \leq \text{cutoff} \} \)
  2. use sequential retrieval on \( M_k \)
- Examples for predicate fac
  - partial equality: \( fac(c_i) \) iff. \( q \) and \( c_i \) are identical w.r.t. at least one attribute
  - local similarity: \( fac(c_i) \) iff. \( q \) and \( c_i \) are sufficiently similar w.r.t. to each attribute
  - partial local similarity: \( fac(c_i) \) iff. \( q \) and \( c_i \) are sufficiently similar w.r.t. to at least one attribute
- Advantage: good performance \( |M_k| \) if small
- Drawbacks
  - retrieval errors may occur \( \Rightarrow \) \( \alpha \)-error: A case \( c_i \) that is sufficiently similar to \( q \) w.r.t. sim is not found (because not considered during preselection).
  - completeness of retrieval is not guaranteed
  - determination of an adequate predicate for preselection is usually difficult

Definition of kd-Trees

- Input
  - \( k \) ordered domains \( T_1, \ldots, T_k \) for attributes \( A_1, \ldots, A_k \)
  - case base \( CB = T_1 \times \ldots \times T_k \)
  - parameter \( b \) (bucket size)
- Definition: A \( kd \)-Tree \( T(CB) \) for case base \( CB \) is a binary tree, that is defined as
  - if \( CB \sim b: T(CB) \) is a leaf of the tree (called bucket), denoted \( CB \)
  - if \( CB > b: T(CB) \) is a tree whose
    - root is denoted with an attribute \( A \) and a value \( v_0 \), \( T_1 \)
    - two sub-trees \( T_1(CB) \) and \( T_2(CB) \) are \( kd \)-trees, too, with
      - \( CB = \{ (x_1, x_2) \in CB \mid x_1 < v_0 \} \) and
      - \( CB = \{ (x_1, x_2) \in CB \mid x_1 > v_0 \} \)

Sequential Retrieval

- Retrieval Task
  - Input
    - case base \( CB = \{ c_1, \ldots, c_n \} \)
    - similarity measure \( sim \)
    - query (new problem) \( q \)
  - Output
    - 1. most similar case \( c_i \)
      - or
    - m most similar cases \( \{ c_{i_1}, \ldots, c_{i_m} \} \)
      - or
    - 3. all cases \( \{ c_1, \ldots, c_n \} \) which have at least a similarity of \( sim_{min} \) to \( q \)
- Main Problem: Efficiency
  - Question: How can the case base be organised in such a way to support an efficient retrieval?

Case Retrieval with kd-Trees

- Index-Oriented Retrieval Procedures
  - preprocessing: generating an index structure
  - retrieval: exploiting the index structure to efficiently access the cases
- Possible Index Structure: \( kd \)-Tree
  - A \( kd \)-Tree is a \( k \)-dimensional binary search tree to support an efficient search over data sets.
  - Idea: partitioning of the data (here: the case base) into small intervals.
  - ordering within a binary tree (similar to a decision tree)
  - during retrieval
    - sorting through the tree from root to the leaves
    - backtracking is possible (unlike in decision trees)

Properties of \( kd \)-Trees

- \( kd \)-tree partitions the case base
  - root represents the entire case base
  - a leaf (bucket) represents a subset of the case base that does not have to be further partitioned
  - at each inner node the case base is partitioned, being divided on the basis of some specific value of an attribute
- Example:
Generating kd-Trees

**Algorithm**

**PROCEDURE CreateTree(CB):** kd-Tree

if |CB|<b then
return leaf node marked with case base CB
else
A_i := choose_attribute(CB)
V_i := choose_split_value(CB,A_i)
return
tree whose root is marked with A_i and V_i
and which has sub-trees
CreateTree( [(x_1,...,x_n)eCB | x_i<=V_i] )
CreateTree( [(x_1,...,x_n)eCB | x_i>V_i] )

**Discussion of kd-Tree Retrieval (I)**

- Retrieval using kd-trees guarantees finding the m nearest neighbours, if the similarity measure used fulfills the following condition:
  Compatibility with ordering and monotony:
  \[
  \forall x_1,...,x_n \text{ and } k',k\text{' such that } x_1<k_1 < k' \text{ and } x_n<k_n < k'_n
  \Rightarrow \text{ sim}(x_1,...,x_n), (x_1,...,x_n) \geq \text{ sim}(x_1,...,x_n), (x_1,...,x_n)
  \]

- Advantages
  - efficient retrieval
  - significant savings in lower dimensions
  - due to the tree structure at least O(log n) operations (comparisons) must be made
  - best case: the similarity between the query and only one case must be calculated
  - effort depends on the number m of most similar cases to find
  - incremental extension of the kd-tree is possible

**Attribute Selection and Splitting Values**

- various methods usable for attribute selection
  - entropy-based
  - inter-quartile distance
  \( \Rightarrow \) choose the attribute with the biggest inter-quartile distance \( \text{iqd} \)

- determination of splitting values
  - median splitting: choose median as splitting value
  - maximum splitting: search for the "largest gap"

**BOB- and BWB-Tests**

- **BOB-Test:** Can there be in the neighbouring sub-tree – any more similar cases (to query q) than the m most similar cases already found?
- **BWB-Test:** Is it guaranteed that there is no case in a neighbouring sub-tree which is more similar to the query q than the m most similar case found so far?

**Discussion of kd-Tree Retrieval (II)**

- **Drawbacks**
  - higher costs for building up the index structure (kd-tree)
  - restrictions implied by kd-trees
    - usability for ordered domains only
    - unknown attribute values are difficult to handle
    - only for monotonous similarity measures that are compatible with the ordering of the respective attribute’s domain
  - dimensionality of the problem is critical
    - in higher dimensions, often the similarity to very many (or even all) – like in linear retrieval cases – must be calculated
    - reason: in higher dimensions, there is the tendency that a query has nearly the same similarity to very many cases; thus the BOB test has to be applied more frequently

- **Further Developments**
  - R-Tree (Guttman et al.), R*-Tree (Kriegel et al.)
Other Retrieval Methods

- Several further advanced retrieval approaches
  - high efficiency
  - general usability depends on problem setting (e.g. case modelling)
- Examples
  - Case Retrieval Nets [Burhardt & Lenz]
  - Retrieval with “Fish and Shrink” [Schaaf, 1994]
  - Case Retrieval on Top of Relational Databases Utilising SQL [Schumacher, 2000]

Part 2: Outlook

5. SOLUTION ADAPTATION
   How to adapt existing solutions to be applicable for the problem at hand?

6. LEARNING IN CASE-BASED REASONING
   Where are the explicit links between CBR and Machine Learning?

7. APPLICATIONS AND TOOLS
   Is CBR actually employed in practice? Are there tools available I may use for trying out some of the things introduced in this talk?

8. REFERENCES
   Where can I find more about CBR?