Vorlesung

Fallbasiertes Schließen

Prof. Dr. Klaus-Dieter Althoff

Raum A8 Spl
Email: althoff@iis.uni-hildesheim.de

Intelligente Informationssysteme

Stand: 22.03.2010

Universität Hildesheim Stiftung 2003

Inhaltsübersicht

GIRLS WILL BE GIRLS

The Girlgroup Musical – Wiederaufnahme – English Drama Group füllt musikhistorische Lücke

- „Die weiblichen Beatles, was die Realität in den Swinging Sixties den Fans bedauerlicherweise vorenthielt, präsentiert die English Drama Group der Universität Hildesheim nun als großes englischsprachiges Comedy-Musical. „Girls will be Girls erzählt die fiktive, aber durchaus realistische Geschichte der Mädchenband „The Lovebirds. Auf der Höhe ihres Erfolges werden Lisa, Sammie, Rickie und Sally vor ein Problem gestellt. Ein verfängliches Foto könnte sie in unangenehmer Weise in die Schlagzeilen bringen. Das würde den Abstieg der gesamten Band bedeuten und muss unbedingt verhindert werden.

  „Das gedankliche Experiment des Stücks war: Wir verlegen die *No Angels* in die Sechziger zurück und schauen, was passiert“, erklärt Paul Willin, Regisseur des Musicals. „Dabei haben wir ein *missing link*, ein fehlendes Glied in der Evolutionsgeschichte der Popmusik hinzuerfunden. Denn eigentlich gab es während der Beatwelle in England keine erfolgreichen Girlgroups“, erläutert Matthias Müller, musikalischer Leiter des Projekts. „Es gab zwar schon vorher im Swing die *Andrew Sisters*, im Doo Wop die *Chordettes* und im Motown die *Supremes*, aber da fehlt doch was dachten wir uns. Die Beatles hatten bisher eben keine Schwestern jetzt haben sie welche!“

  Die gelungene Show wurde vor neun Jahren in Hildesheim uraufgeführt und erhielt nun ein *Update*, denn es ist nicht nur die aufwendigste und erfolgreichste Produktion in der Geschichte der English Drama Group, sondern auch das erste selbst geschriebene Musical. Der Autor des Stücks, Francis Jarman, hat im Laufe der letzten zehn Jahre eine ganze Reihe von Theaterstücken für die English Drama Group verfasst und für dieses komponierten Paul Harrison, Paul Willin und Matthias Müller hitverdächtige Songs im Stil und im Sound des britischen 60s Beat angereichert mit Girl Power.

  Aufführungen von **Girls will Be Girls**: **22.-26. Juni 2009 um 20 Uhr im Audimax der Universität Hildesheim**, Marienburger Platz 22, Hildesheim.

§ 5 Effizientes Retrieval

- Sequentielles Retrieval
- Zweistufiges Retrieval
  - Relationales Retrieval (mit DB)
- Indexbasiert
  - Retrieval mit kd-Bäumen
  - Retrieval mit Netzen (CRN)
  - Retrieval mit Fish and Shrink

Inhaltsverzeichnis

Effizientes Retrieval von Fällen

- Zentrale Aufgabe des Retrievals:
  - gegeben:
    - Fallbasis FB = $\{F_1,...,F_n\}$ und Ähnlichkeitsmaß sim
    - Anfrage: Q (neues Problem)
  - gesucht entweder:
    - 1. der ähnlichste Fall F_i ODER
    - 2. die m ähnlichsten Fälle $\{F_1,...,F_m\}$ (geordnet oder ungeordnet) ODER
    - 3. alle Fälle $\{F_1,...,F_m\}$, die zur Anfrage Q mindestens die Ähnlichkeit sim_{min} haben
- Hauptproblem:
  - Effizienz
- Frage:
  - Wie organisiert man die Fallbasis, so dass Fälle effizient aufgefunden werden können?

Sequentielles Retrieval

- Datenstrukturen (auch für kd-Bäume verwendet)
  - Typen:

```
SimCase = RECORD
            case: Fall;
            similarity: [0..1]
          END;
SimCaseQueue = ARRAY[1..m] OF SimCase; //absteigend sortiert nach similarity
```

  - Variablen:

```
scq: SimCaseQueue          /* m ähnlichsten Fälle */
cb: ARRAY [1..n] OF Fall   /* Fallbasis */
```

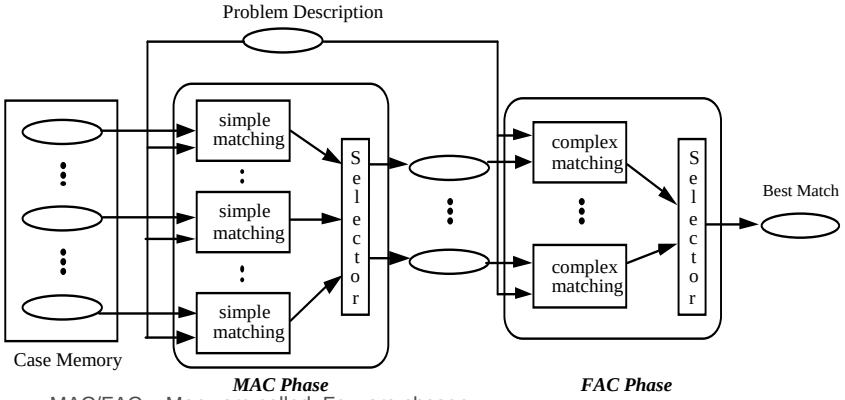
- Retrieval-Algorithmus

```
FOR i:= 1 TO m DO scq[i].similarity := 0
FOR i:= 1 TO n DO
   IF sim( Q, cb[i] ) >  scq[m].similarity THEN
         füge cb[i] in scq ein (sortiert)
RETURN scq
```


Sequentielles Retrieval – Eigenschaften

- Komplexität des sequentielles Retrieval: O(n)
- Vorteile:
  - Einfache Implementierung
  - Es müssen keine Indexstrukturen aufgebaut und erweitert werden
  - Es können beliebige Ähnlichkeitsmaße verwendet werden
- Nachteile:
  - Trotz niedriger Komplexitätsklasse problematisch, wenn Fallbasis sehr groß
  - Retrievalaufwand ist unabhängig von der Anfrage
  - Retrievalaufwand ist unabhängig von m

Zweistufiges Retrieval: MAC/FAC-Modell



- MAC/FAC = Many are called; Few are chosen
- http://www.qrg.northwestern.edu/ideas/macfac.htm
- Gentner, D. and Forbus, K. (1991). MAC/FAC: A model of similarity-based retrieval.

Zweistufiges Retrieval

- Idee:
  - zweistufiges Verfahren
    - **Mac/Fac** (Many are called; Few are chosen)
      1. Vorauswahl von möglichen Lösungskandidaten $M_Q \subseteq FB$
         - » $M_Q = \{F \in FB \mid SIM(Q,F)\}$
      2. Anordnung der Lösungskandidaten nach Ähnlichkeit **sim**
         - » Anwendung des sequentiellen Retrievals
- Problem:
  - Finden des Prädikats SIM

Beispiele für Prädikate zur Vorauswahl

- Partielle Gleichheit:
  - SIM(Q,F) **gdw.**
    Q und F stimmen in mindestens einem Attribut überein
- Lokale Ähnlichkeit
  - SIM(Q,F) **gdw.**
    Q und F sind bezüglich jedes Attributs hinreichend ähnlich
- Partielle lokale Ähnlichkeit
  - SIM(Q,F) **gdw.**
    Q und F sind bezüglich eines Attributs hinreichend ähnlich

Eigenschaften des zweistufigen Retrievals

- Vorteile:
  - Performanzvorteil, wenn die Vorauswahl wenige Fälle selektiert
- Nachteile:
  - Retrievalfehler sind möglich, d.h.
    - **α-Fehler**:
      - Ein Fall, der zur Anfrage zwar bzgl. **sim** hinreichend ähnlich ist, wird nicht ausgewählt
        - » da durch Vorauswahl nicht berücksichtigt

      → **Vollständigkeit des Retrievals ist nicht gewährleistet !**
  - Bestimmung eines geeigneten Ähnlichkeitsprädikats SIM ist in der Regel schwierig

Relationales Retrieval (1)

- Vorauswahl

  Bei der Vorauswahl wird zunächst aus der Fallbasis CB eine Teilmenge M von möglichen Lösungskandidaten bestimmt.
  - Durch eine oder mehrere Anfragen an die (rel.) Datenbank
  - Die Vorauswahl schließt nicht für die Lösungsfindung geeignete Fälle aus, d.h. sie realisiert im Idealfall eine Art von binärem Filter, den nur hinreichend ähnliche passieren können.
- Anordnung der Alternativen

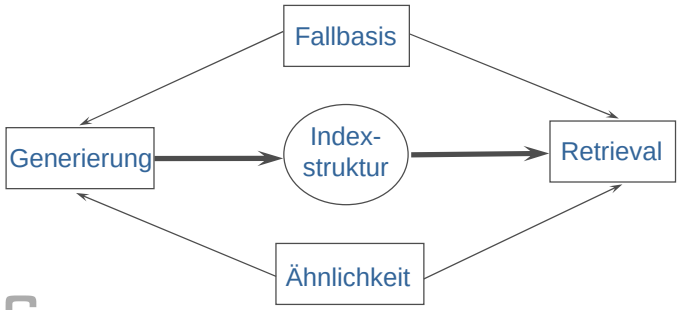
  Die in der ersten Phase gefundenen Kandidaten werden durch die sequentielle Berechnung des vorgegebenen Ähnlichkeitsmaßes entsprechend ihrer Ähnlichkeit – und damit hoffentlich entsprechend ihrer Eignung für die Lösung der aktuellen Anfrage Q – angeordnet.

Relationales Retrieval (2)

- Vorteile
  - Möglicher Performanzvorteil
  - Korrektheit des Verfahrens
  - Retrievalaufwand ist abhängig von der Art der Anfrage
  - Nutzung von Wissen zur Beschränkung des Aufwandes
  - Adhoc-Anfragen beschränkt möglich
- Nachteile
  - Mgl. Retrievalfehler bleibt
  - Beschränkung der Ähnlichkeitsmaße
  - Performanzvorteil ist nicht gesichert
  - Bestimmung von Queries für die Vorauswahl ist i.A. schwierig

Indexorientierte Retrievalverfahren

- Preprocessing:
  - Generierung einer Indexstruktur
- Retrieval:
  - Verwenden der Indexstruktur zum effizienten Zugriff auf Fälle



Retrieval mit kd-Bäumen
(**Wess 1995**: im CBR-Bereich)

- k-dimensionaler binärer Suchbaum zur effizienten Suche in Datensätzen
  - Bentley, 1975
- Idee:
  - Zerteilung des Datensatzes (hier: Fallbasis) in immer kleinere Intervalle
- Anordnung in einem binären Baum
  - ähnlich zu einem Entscheidungsbaum
- Beim Retrieval:
  - Durchlaufen des Baumes bis zu einem Blattknoten
  - Im Gegensatz zum Entscheidungsbaum ist Backtracking jedoch möglich.

Definition: kd-Baum

- Gegeben seien
  - k geordnete Wertebereiche $T_1,...,T_k$ der Attribute $A_1,...,A_k$,
  - eine Fallbasis $FB \subseteq T_1 \times ... \times T_k$ und
  - ein Parameter b (Bucketgröße).
- Ein **kd-Baum** T(FB) für die Fallbasis FB ist ein binärer Baum, der wie folgt definiert ist:
  - Ist $|FB| \leq b$: T(FB) ist ein Blattknoten (sog. Bucket), der mit FB markiert ist.
  - Ist $|FB| > b$: T(FB) ist ein Baum, dessen
    - Wurzel, die mit einem Attribut A_i und einem Wert $v_i \in T_i$ markiert ist und
    - die zwei kd-Bäume $T_{\leq}$ ($FB_{\leq}$) und $T_{>}$($FB_{>}$) als Nachfolger besitzt, wobei $FB_{\leq} := \{(x_1,...,x_k) \in FB \mid x_i \leq v_i\}$ und $FB_{>} := \{(x_1,...,x_k) \in FB \mid x_i > v_i\}$

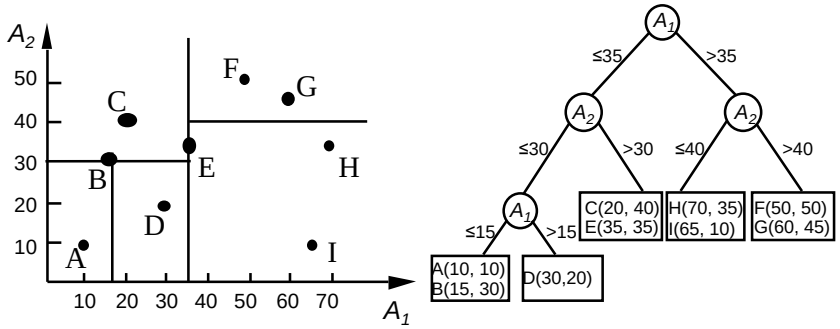
Eigenschaften des kd-Baum

- Ein kd-Baum **partitioniert** eine Fallbasis:
  - die Wurzel repräsentiert die gesamte Fallbasis
  - ein Blattknoten (Bucket) repräsentiert eine Teilmenge der Fallbasis, die nicht weiter partitioniert werden soll
  - bei jedem inneren Knoten wird die Fallbasis weiter partitioniert,
    - wobei die Fallbasis bzgl. eines Wertes eines Attributs geteilt wird.

Beispiel für einen kd-Baum

FB={A,B,C,D,E,F,G,H,I}

Wie groß ist hier die Bucketgröße b?



Generierung von kd-Bäumen (1)

Algorithmus:

PROCEDURE CreateTree(FB): kd-tree

IF $|FB| \leq b$

THEN RETURN Blattknoten mit Fallbasis FB

ELSE

A_i := wähle_Attribut(FB);

v_i := wähle_Wert(FB,A_i)

RETURN

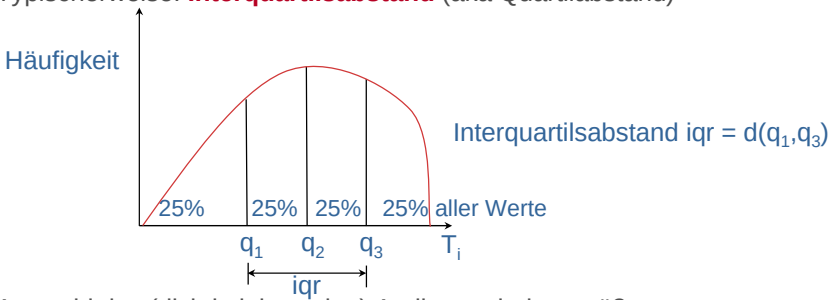
Baum mit der Wurzel, die mit A_i und v_i markiert ist, und die beiden Teilbäume

CreateTree($\{(x_1,...,x_k) \in FB \mid x_i \leq v_i\}$) und

CreateTree($\{(x_1,...,x_k) \in FB \mid x_i > v_i\}$) enthält.

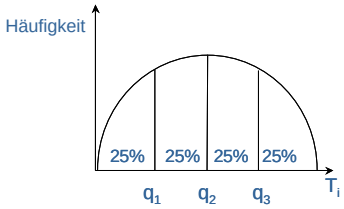
Attributauswahl A_i

- Viele Verfahren einsetzbar, z.B. auch Entropie
- Typischerweise: **Interquartilsabstand** (aka Quartilabstand)

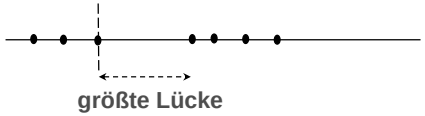


- Auswahl des (diskriminierenden) Attributs mit dem größten Interquartilsabstand (dabei haben die entsprechenden Quartile die kleinste lokale Ähnlichkeit, was zu höchstem Abstand korrespondiert)

Wertauswahl v_i



- Zwei Verfahren:
  - Mediansplitting:
    - wähle den Median q_2 als Partitionswert
  - Maximumsplitting:



    - Aufteilung in zwei Hälften, wo Abstand zwischen größtem Wert aus linker Hälfte und kleinstem Wert aus rechter Hälfte maximal ist.

Retrieval mit kd-Bäumen (1)

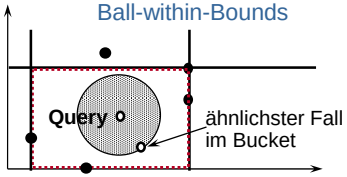
Idee für einen Algorithmus:

1. Durchlaufe den Baum bis zu einem Blatt
2. Berechne Ähnlichkeit zu gefundenen Fällen
3. Bestimme zusätzliche Kandidaten durch einen BOB-Test
4. Bestimme durch BWB-Test, ob Terminierung

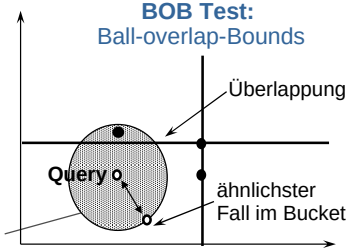
4a. Wenn „überlappende Buckets“ existieren, dann durchsuche alternative Zweige (Backtracking zu Schritt 3)

4b. Stop, wenn keine „überlappenden Buckets“

BWB Test
Ball-within-Bounds



BOB Test:
Ball-overlap-Bounds

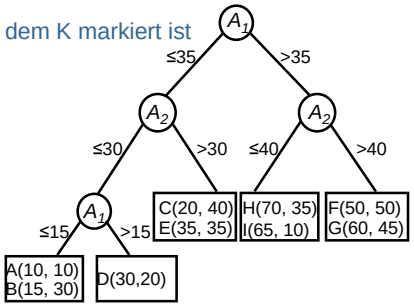


„Geometrisches“ Ähnlichkeitsmaß korrespondierend zum Euklidschen Abstand mit k Dimensionen, z.B. für k = 2

Retrieval mit kd-Bäumen (2) – Algorithmus

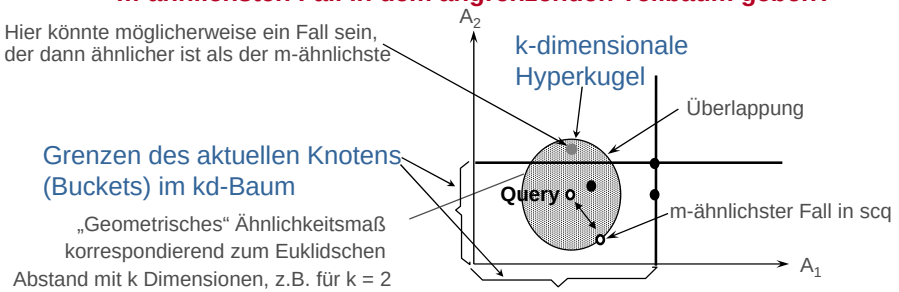
```
PROCEDURE Retrieve(K: kd-baum)
  IF K ist Blattknoten THEN
    FOR jeden Fall F von K DO
      IF sim(Q,F) >  scq[m].similarity THEN füge F in scq ein
  ELSE  /* innerer Knoten */
    Sei Ai das Attribut und vi der Wert, mit dem K markiert ist
    IF Q[Ai] ≤ vi THEN
      Retrieve(K≤)
      IF BOB-Test ist erfüllt THEN
        Retrieve(K>)
    ELSE /* Q[Ai] > vi */
      Retrieve(K>)
      IF BOB-Test ist erfüllt THEN
        Retrieve(K≤)
  IF BWB-Test ist erfüllt THEN terminiere Retrieval mit scq
  ELSE RETURN to parent node
```

a priority list scq is maintained which contains the m most similar cases known so far, together with their similarity to the problem case Q.



BOB-Test
- Ball-Overlap-Bounds -

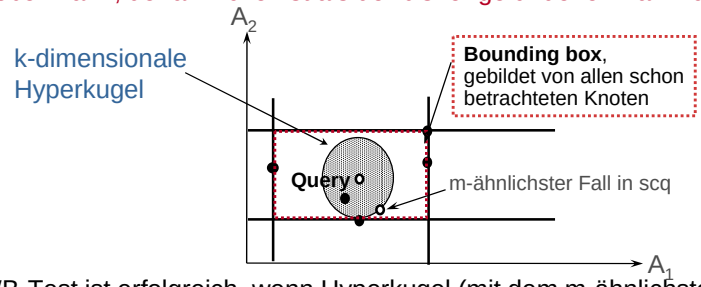
Kann es ähnlichere Fälle als den bisher gefundenen m-ähnlichsten Fall in dem angrenzenden Teilbaum geben?



Wenn (Hyper-)kugel eine Bucket-Grenze schneidet, können in diesem anderem Bucket (hier: darüber) potentiell auch Fälle sein, die eine höhere sim als der m-ähnlichste Fall haben → Bucket (darüber) auch betrachten

BWB-Test
- Ball-Within-Bounds -

Ist sichergestellt, dass es keinen Fall in dem angrenzenden Teilbaum geben kann, der ähnlicher ist als der bisher gefundene m-ähnlichste Fall?



BWB-Test ist erfolgreich, wenn Hyperkugel (mit dem m-ähnlichsten Fall) um die Query in bounding box (Raum aller untersuchten Buckets) ist, es also keine Überschneidung gibt.

Textuelle Beschreibung zu BWB- und BOB-Test

Ist man also mit Hilfe des Anfragefalls in einem Bucket angekommen, so ist vor allen Dingen erst einmal nötig festzustellen, ob man die n ähnlichsten Fälle überhaupt schon gefunden hat oder vielleicht nur einen Teil von n oder u. U. auch gar keinen der n ähnlichsten Fälle. Dies wird mit dem sogenannten *Ball-Within-Bounds*-Test, kurz BWB-Test, geprüft. Dieser Test resultiert aus einer geometrischen Vorstellung des Datenraums $\mathcal{W}$ (s. Definition 2.1), in dem man sich die Fälle als Punkte vorstellen kann. Nun denkt man sich den Anfragefall als Mittelpunkt einer k-dimensionalen Hyperkugel mit dem Radius r. Wählt man nun den Radius r so[6], daß er dem zur Zeit n-größten Ähnlichkeitswert entspricht, so muß die in dieser Weise konstruierte Kugel alle Punkte, sprich Fälle, enthalten, welche einen mindestens genauso großen Ähnlichkeitswert zum Anfragefall haben. Betrachtet man die durch den k-d-Baum herleitbaren geometrischen Beschreibungen des Datenraums, den ein Bucket repräsentiert, so kann man testen, ob außerhalb des betreffenden Buckets kein ähnlicherer als der bisher n-ähnlichste Fall gefunden werden kann. Dies ist nämlich genau dann der Fall, wenn die Hyperkugel vollständig innerhalb der Bucketbeschreibung liegt.

innerhalb dieses Teilbaumes auch wirklich bessere, d. h. zum Anfragefall ähnlichere Fälle befinden können, wird jetzt der zweite Test, der sogenannte *Bounds-Overlap-Ball*-Test, kurz BOB-Test, angewendet. Dieser bildet analog zum BWB-Test die aktuelle Hyperkugel und untersucht, ob diese die zum betrachteten Teilbaum gehörige Beschreibung des Datenraums schneidet, wie das z. B. in Abbildung 2.7 der Fall ist.

[Derwand2004, S. 19]

Einschränkung der verwendbaren Ähnlichkeitsmaße

- Das Retrieval mit einem kd-Baum garantiert das Finden der m-nächsten Nachbarn, wenn das verwendete Ähnlichkeitsmaß folgenden Bedingungen genügt:
  - Verträglichkeit mit der Ordnung und Monotonie:
    - $\forall\ x_1,...,\ x_n,\ x_i',\ x_i''$ wenn $x_i <_i x_i' <_i x_i''$ dann $sim(\ (x_1,...,x_n),\ (x_1,...,x_i',...,x_n)\ ) \geq sim(\ (x_1,...,x_n)\ ,\ (x_1,...,x_i'',...,x_n)\ )$

Eigenschaften des Retrievals mit kd-Bäumen

- Nachteile:
  - Erhöhte Kosten zum Aufbau der Indexstruktur
  - Beschränkungen des kd-Baums:
    - nur für geordnete Wertebereiche
    - Probleme mit unbekannten Werten
    - Nur für monotone Ähnlichkeitsmaße, die mit Ordnung verträglich sind
- Vorteile:
  - Effizientes Retrieval
  - Aufwand hängt von der Anzahl **m** zu findender Fälle ab
  - Inkrementelles Erweitern beim Auftreten neuer Fälle ist möglich
  - Speichern von Fällen in Datenbank ist möglich
- Verbesserungen von kd-Bäumen sind entwickelt worden
  - Z.B. INRECA Tree für ungeordnete und unbekannte Werte EU-Projekt INRECA (Wess 1995; Althoff 1997)

Case-Retrieval-Netze (CRN) (Lenz & Burkhard 1996)

- Aufbau eines Netzes (vgl. Neuronales Netz) zwecks Zugriff
- Zerlegen der Fallinformationen in Informationseinheiten (IEs) (z.B. Attribut-Wert-Paar)
  - Jede Informationseinheit wird ein Netzknoten
  - Jeder Fall wird durch einen Fallknoten repräsentiert
- Informationseinheiten, zwischen denen eine Ähnlichkeit >0 besteht, werden verbunden.
  - Verbindungsstärke = Ähnlichkeit.
- Zum Retrieval werden die Informationseinheiten der Anfrage aktiviert.
- Die Aktivität wird durch das Netzwerk bis zu den Fallknoten propagiert.
- Aktivität an den Fallknoten spiegelt die Ähnlichkeit zur Anfrage wider.

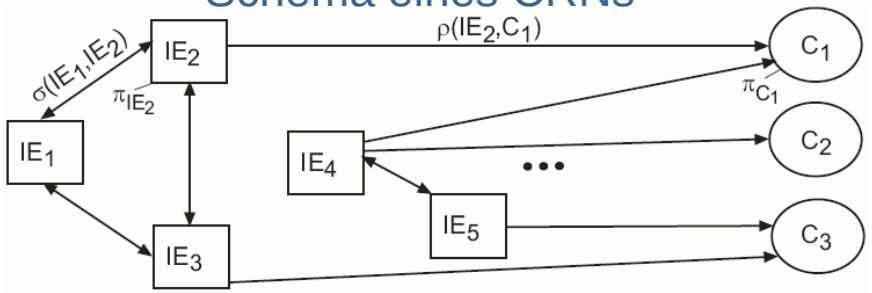
CRN-Begriffe (1)

- Grundlegend für das CRN ist: **Eine Informationseinheit (IE) ist eine atomare Wissenseinheit**
  - z.B. Attribut-Wert-Paar
  - Für numerische Werte mehrere IEs
  - Text: Pro Stammform ein IE
- Ein Fall wird durch eine Menge von Informationseinheiten repräsentiert
  - z.B. AVPs, oo-Repräsentationen, Texte
- Eine Query (Anfrage) besteht auch aus einer Menge von Informationseinheiten

Definition CRN

- Ein Retrieval-Netz (Basic Case Retrieval Net, BCRN) ist ein 5-Tupel $N=(E, C, \sigma, \rho, \Pi)$ mit:
  - E ist eine endliche **Menge von Informationseinheiten**
  - C ist eine endliche **Menge von Fallknoten** (auch Falldeskriptoren)
  - σ ist eine **Ähnlichkeitsfunktion**: $\sigma: E \times E \rightarrow IR$
    - $\sigma(e_i, e_k)$ beschreibt die lokale Ähnlichkeit zwischen zwei IEs e_i, e_k
  - ρ ist eine **Relevanzfunktion**: $\rho: E \times C \rightarrow IR$
    - $\rho(e,c)$ beschreibt die Relevanz (Gewicht) des IEs e für den Fall c
  - Π ist eine endliche **Menge von Propagierungsfunktionen** π_n: $IR^E \rightarrow IR$ für jeden Knoten $n \in E \cup C$ (vgl. Amalgamierungsfunktion).

Schema eines CRNs

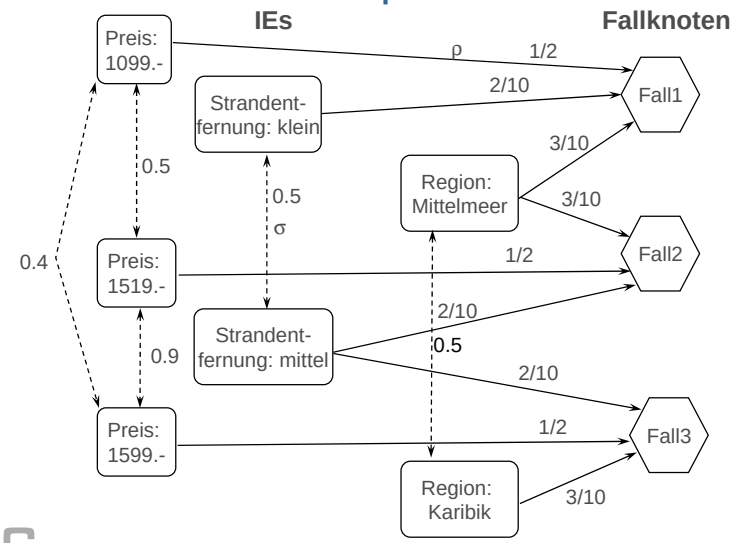


- gerichteter Graph
- zwei Arten von Knoten: IE-Knoten und Falldeskriptoren
- zwei verschiedene, gewichtete Kantentypen
  - Bidirektionale Ähnlichkeitskanten drücken die je nach Typ (der IEs) gewichtete Ähnlichkeit zwischen zwei IEs aus. $\sigma(IE_1,IE_2)$
  - Relevanzkanten von IEs zu den Falldekriptoren bestimmen die Relevanz dieser IEs für einen Fall $\rho(IE_2,C_1)$
- Propagierungsfunktionen der IE- bzw. Fallknoten lauten π_{IE_i} bzw. π_{C_i}

Fälle im CRN

- **Eine Fallrepräsentation besteht daher aus einem Teilgraph**, mit einem Fallknoten und allen IE-Knoten,
  - mit denen er über Relevanzkanten verbunden ist, welche ihn repräsentieren.
- Ähnlichkeiten zwischen den IEs als auch die IEs selbst reflektieren dabei die Eigenschaften der Domäne
  - [Richter2003, S. 419].
- Propagierungsfunktion eines Falldeskriptors berechnet die globale Ähnlichkeit,
  - für AVPs ist dies eine Aggregationsfunktion

Beispiel



Aufbau eines CRN

geg : *CRN* Die CRN – Struktur
for (alle $C \in$ *Fallbasis*(*CB*))
  Füge *Fallknoten c* für *C* in *CRN ein*
  for (alle *IEs* in *C*)
    if (*CRN* enthält *dieses IE* noch nicht) *then*
      Füge *IE e* dem *CRN* hinzu
    end if
    füge Verweis von *e* auf *Fallknoten c* hinzu (Relevanzkante)
  end for
end for
for (*alle IEs der Fallbasis*)
  Ziehe bidirektionale Ähnlichkeitskanten zwischen allen IEs, die zu diesem Attribut gehören
  berechne in diesem Schritt auch die lokalen Ähnlichkeiten
end for

Begriffe (2)

- Eine **Aktivierung** eines BCRN ist eine Funktion α: E ∪ C → IR.
- Die **Aktivierung α(e) eines IE e drückt die Bedeutung dieses IEs für das aktuelle Problem aus**.
- Die **Aktivierung zum Zeitpunkt t+1** ist eine Funktion α_t: E ∪ C → IR, die folgendermaßen bestimmt ist:
  - IEs: $\alpha_{t+1}(e) = \pi_e(\sigma(e_1,e)\cdot\alpha_t(e_1),...,\sigma(e_s,e)\cdot\alpha_t(e_s))$
  - Fälle: $\alpha_{t+1}(c) = \pi_c(\rho(e_1,c)\cdot\alpha_t(e_1),...,\rho(e_s,c)\cdot\alpha_t(e_s))$

Retrieval im CRN

- Gegeben: Query Q, bestehend aus einer Teilmenge der IEs
- Retrieval erfolgt in 3 Schritten:
  1. **Initiale Aktivierung**: Bestimmung der Aktivierung α_0 durch:
  $$\alpha_0(e) = \begin{cases} 1 & \text{falls IE e kommt in der Query vor} \\ 0 & \text{sonst} \end{cases}$$
  2. **Ähnlichkeitspropagierung**: für alle e ∈ E, die (mind.) eine Verbindung zu einem aktivierten IE haben (entlang der Ähnlichkeitskanten).
  $$\alpha_1(e) = \pi_e(\sigma(e_1,e)\cdot\alpha_0(e_1),...,\sigma(e_s,e)\cdot\alpha_0(e_s))$$
  3. **Relevanzpropagierung**: für alle Fallknoten c ∈ C, die eine Verbindung zu einem aktivierten IE haben (entlang der Relevanzkanten).
  $$\alpha_2(c) = \pi_c(\rho(e_1,c)\cdot\alpha_1(e_1),..., \rho(e_s,c)\cdot\alpha_1(e_s))$$
- Ergebnis: **Aktivierung der Fälle reflektiert Ähnlichkeit zur Query**!

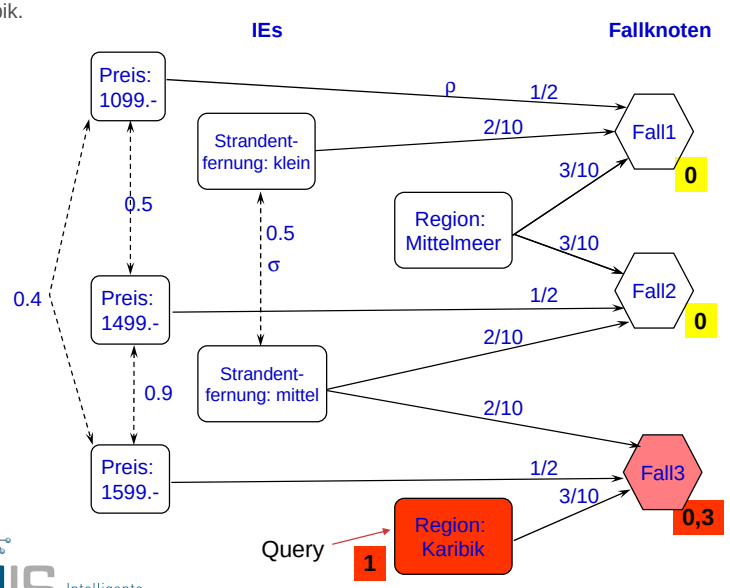
Retrieval-Algorithmus in Pseudocode

Input: Case Retrieval Net (**E**,**C**, *σ*, *ρ*, *π*) and query *q*.*
Output: Sorted list of cases
procedure Retrieve((**E**,**C**, *σ*, *ρ*, *π*), *q*)
var
  α : **array** [1..|**E** ∪ **C**|] **of** [0, 1]
begin
  for each IE∈ **E** ∪ **C** **do** *α*[IE] := 0 **// 1. Schritt**
  for each IE∈ *q* **do** *α*[IE] := 1
  for each IE∈ **E** such that it exists IE'∈ **E** such that *σ*(IE',IE) > 0 **do**
    α(IE) := $\pi_{IE}(\sigma(IE_1,IE) * \alpha(IE_1), \ldots, \sigma(IE_k,IE) * \alpha(IE_k))$ **// 2. Schritt**
  for each C∈ **C** with it exists IE ∈ **E** such that *ρ*(IE,C) > 0 **do**
    α(C) := $\pi_C(\rho(IE_1,C) * \alpha(IE_1), \ldots, \rho(IE_k,C) * \alpha(IE_k))$ **// 3. Schritt**
  return sort(*C* with *α*(*C*) |*C* ∈ **C** ^ *α*(*C*) > 0)
end

* im Original als problem description d, aus [Bergmann2002, S. 208]

Beispiel (1)

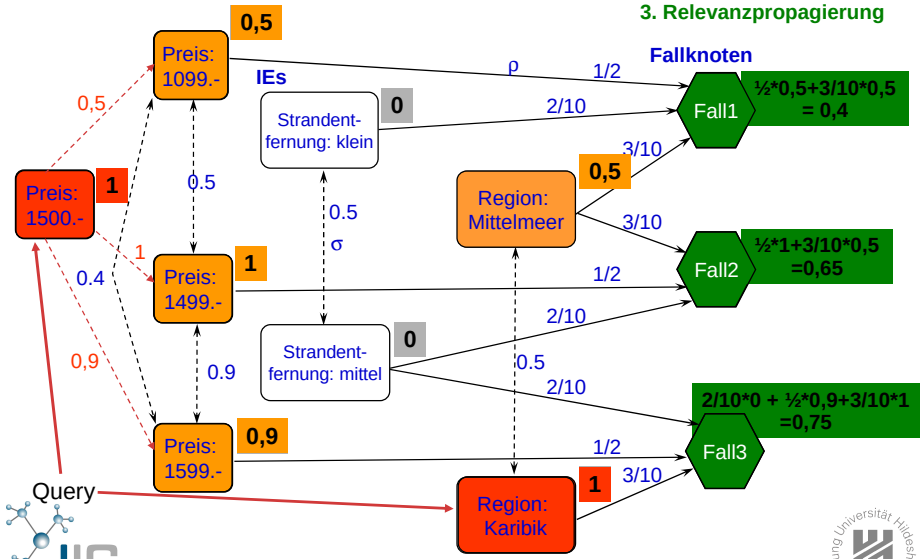
Query:
Finde ein Urlaubsziel in der Karibik.



Beispiel (2)

Query:
Finde ein Urlaubsziel in der Karibik mit Preis = 1500 €.

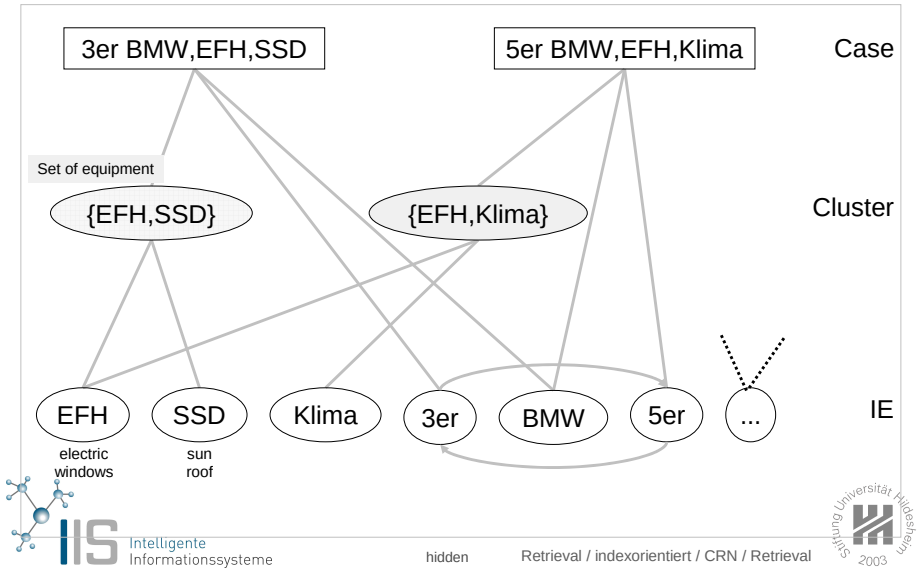
1. Initiale Aktivierung
2. Ähnlichkeitspropagierung
3. Relevanzpropagierung



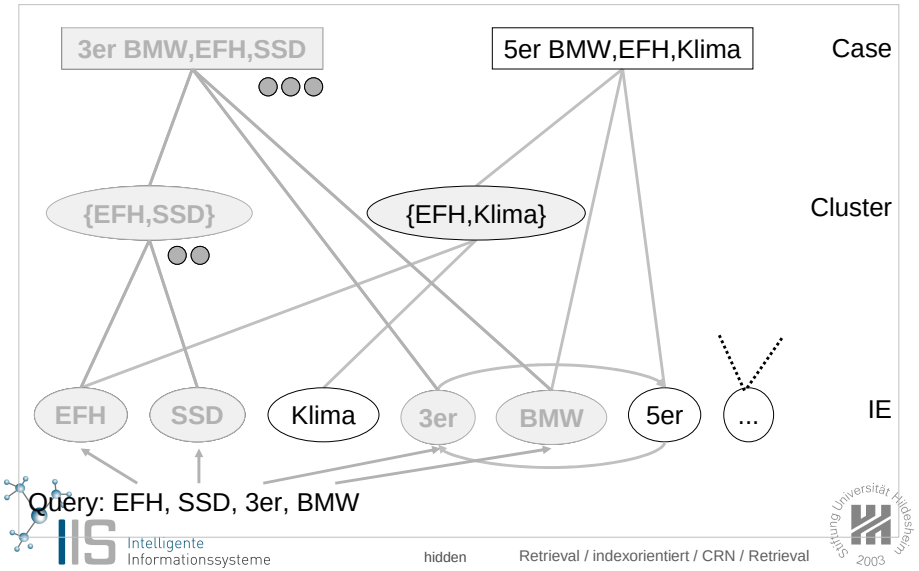
Weiteres Beispiel Aktivierung

- Vielleicht in VL oder Übung verwenden

KS Index – CRN Structure

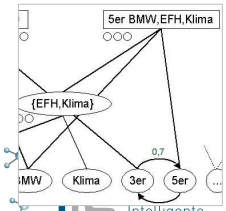


KS Index – Retrieval

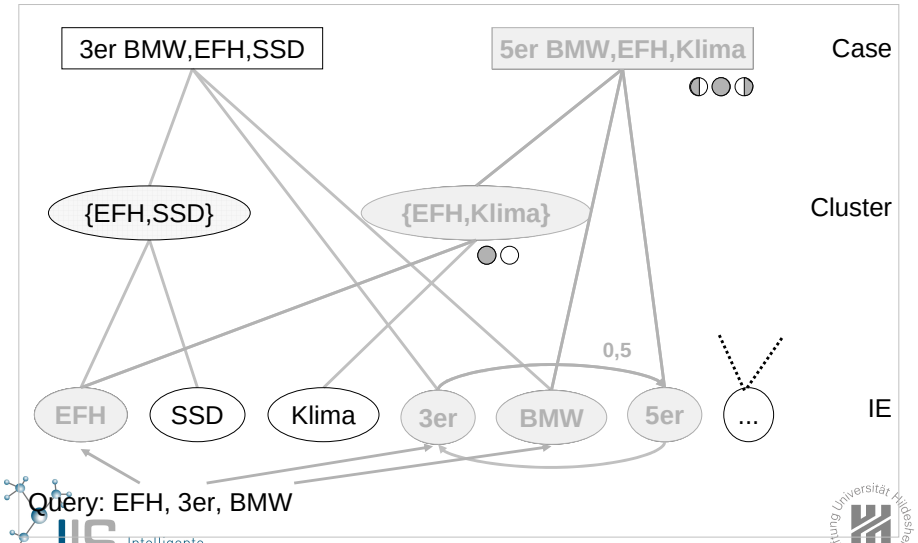


KS Index – Case Inclusion

- Use
  - Similarity measure used for sets
  - Typical use case, i.e. for e-commerce applications to realise a logical OR (a product has only one producer but the customer should have the possibility to select more than one for searching)
- How is it working
  - Query: {a,b,c,d}
  - Case 1: {a} with 100 % similarity
  - Case 2: {b} with 100 % “
  - Case 3: {c,d} with 100 % “
  - Case 4: {a,e} with 50 % “
  - Case 5: {e} with 0 % “

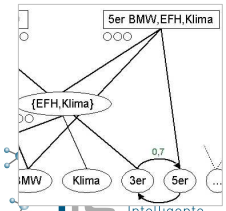


KS Index – Retrieval Case Inclusion

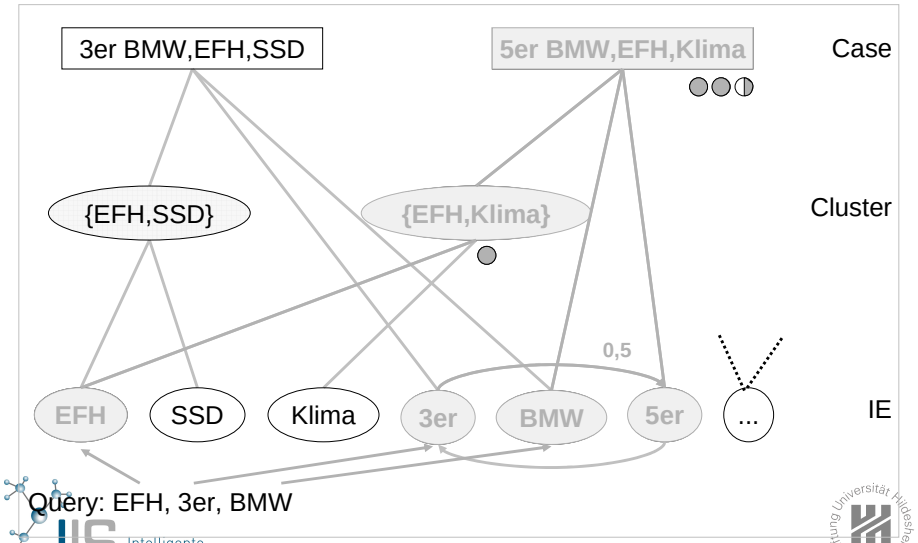


KS Index – Query Inclusion

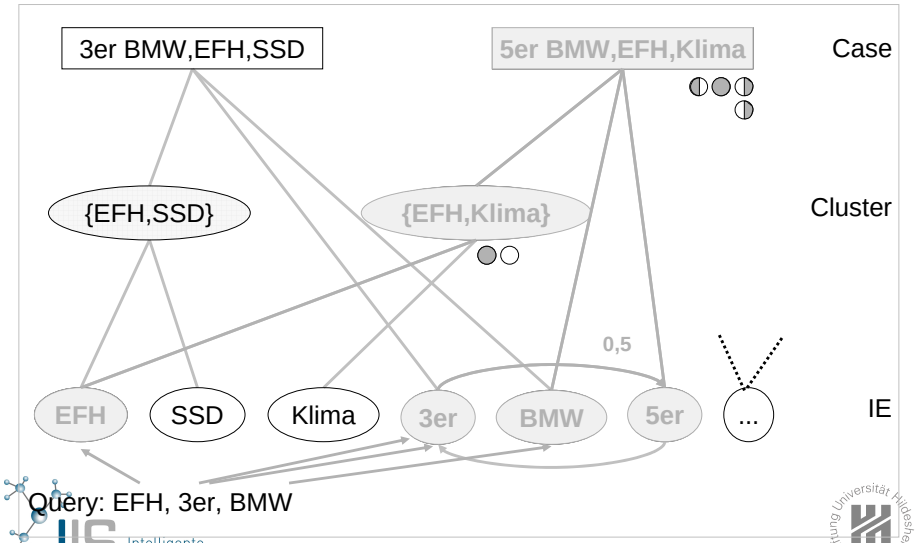
- Use
  - Similarity measure used for sets
  - Typical use case, i.e. KM so that the required document contains as many criteria as possible (realisation of a logical AND)
- How is it working
  - Query: {a,b,c,d}
  - Case 1: {a} with 25 % similarity
  - Case 2: {b} with 25 % “
  - Case 3: {b,d} with 50 % “
  - Case 4: {e} with 0 % “



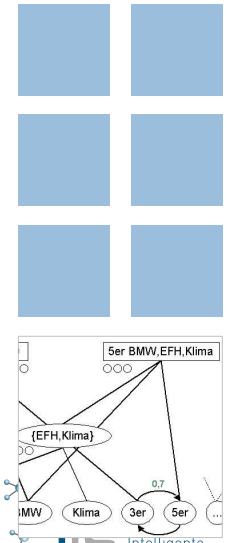
KS Index – Retrieval Query Inclusion



KS Index – Retrieval Weights



orenge:TextMiner – Overview



- Problem: structuring free text entries both for queries and indexing
- Procedure
  - Recognition of concepts based on defined keywords in the knowledge model
  - Recognition of concepts based on pattern and regular expressions
  - Handling of wildcard-expressions
  - Handling of misspelled words
  - Generation of filters based on textual pattern

Eigenschaften der Retrieval-Netze

- Vorteile:
  - Effizientes Retrieval
  - Aufwand hängt von der Anzahl der aktivierten IEs der Query ab
  - Inkrementelles Erweitern beim Auftreten neuer Fälle ist möglich
- Nachteile:
  - Erhöhte Kosten zum Aufbau des Netzes.
  - Bei numerischen Attributen müssen ggf. neue Anfrageknoten erzeugt werden.
  - Bei hohem Vernetzungsgrad (viele IEs sind zueinander ähnlich) müssen viele Propagierungen durchgeführt werden.
- Verbesserungen des Basisansatzes sind entwickelt worden
  - z.B. Lazy Spreading Activation
  - Fast CRNs

Ähnlichkeitsmodellierung für CRN

- IEs in verschiedene Klassen:
  - STOPWORD;Stop-Wörter;0;SW;0.0
  - COMP_SCIENCE;Begriffe aus der IT-Welt;1;TE;1.0
  - GENERAL;allgemeinsprachliche Begriffe;2;TE;1.0
  - AV_PAIR;Attribut-Werte-Paare;4;AV;1.0
  - GASPL2;GASPL2-spezifische Begriffe;5;TE;1.0
- Bsp für IE
  - __ADRESSE__[1];ADRESSE;ADRESSEN;ADRESSIEREN;ADRESSIERT
- Sim-Klassen
  - SIM_ABSTRACTION;0;TS;0.3
  - SIM_ISPART;3;TS;0.5
  - SIM_ISMEANSFOR;5;TS;0.7
- Sims
  - __16-BIT-ADRESSE__[7];__SPEICHER__
  - __HAUS__[0];__HOTEL__

Retrieval mit “Fish and Shrink“
J. W. Schaaf (1998) / Bergmann (2002)

- Fallrepräsentation ist in verschiedene **Aspekte** aufgeteilt.
  - **Aspekt ist eine komplexe Eigenschaft**
    - Beispiel für Aspekte in der Medizin: EKG, Röntgenbild, Anamnese.
  - Für jeden Aspekt sind eigene Ähnlichkeitsmaße definiert.
  - Annahme:
    - Ähnlichkeitsberechnung für Aspekte ist sehr aufwendig.
  - Gesamtähnlichkeit berechnet sich durch gewichtete Kombination (Aggregation) der Aspektähnlichkeiten.
  - Gewichte sind nicht statisch
    - sondern werden durch die Anfrage festgelegt.
- Ansatz für Retrieval:
  - **Vorausberechnung der Aspektähnlichkeiten** zwischen bestimmten Fällen
  - **Aufbau eines Netzwerkes**
  - **Test der Ähnlichkeit** zwischen der Anfrage und einem Testfall
  - **Rückschluss auf Ähnlichkeit** für andere verbundene Fälle möglich
    - ohne Ähnlichkeit berechnen zu müssen

Eigenschaften von Fish and Shrink

- Vorteile:
  - Flexible Distanzberechnung zur Anfrage (Sichten)
  - Verschiedene Retrievalaufgaben können berücksichtigt werden
  - Effizient, da viele Distanzberechnungen eingespart werden können
  - Kann als **Anytime-Algorithmus** eingesetzt werden
- Nachteile:
  - Aspektdistanz zwischen Fällen muss vorab berechnet werden
  - Distanzfunktion muss Dreiecksungleichung erfüllen
- Geeignet, wenn sehr aufwendige Distanzmaße vorliegen
  - z.B. bei Graph-Repräsentationen

Kap. 5: Effizientes Retrieval

Typ	*Method*	*Restriction with respect to Similarity*	*Appropriate for ...*
brute force	**sequential search**	no	small case bases simple similarity
index based	**kd-tree** *(Wess, Althoff, Derwand, 1993)*	reflexivity, monotony no class similarity	small number of attributes large case bases
	Fish & Shrink *(Schaaf, 1996)*	reflexivity, monotony triangle inequality	complex similarity small case bases
	Retrieval Nets *(Burkhard & Lenz, 1996)*	monotony, no class similarity	few numeric attributes large case bases
dyn. database queries	**SQL Approximation** *(Schumacher & Bergmann, 2000)*	monotony no class similarity linear aggregation functions	simple similarity large case bases dynamic case bases

Literatur

- Althoff, K.-D. (1997). *Evaluating Case-Based Reasoning Systems: The Inreca Case Study*. Habilitationsschrift, Fachbereich Informatik, TU Kaiserslautern.
- Althoff, K.-D., Auriol, E., Barletta, R. & Manago, M. (1995). *A Review of Industrial Case-Based Reasoning Tools*. AI Intelligence, United Kingdom
  - http://www.iis.uni-hildesheim.de/~althoff/Publications/Resources/CBR-REVIEW_gesamt.pdf
- Bentley, J.L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18, 509-517
- Lenz, M. (1999). *Case Retrieval Nets as a Model for Building Flexible Information Systems*. DISKI-Reihe Nr. 236, infix, Berlin: Akad. Verl.-Ges. Aka.
- Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D. & Wess, S. (eds.) (1998). *Case-Based Reasoning Technology: From Foundations to Applications.* Lecture Notes in Artificial Intelligence, Vol. 1400, Springer, 405 pages.
- Wess, S. (1995). *Fallbasiertes Schließen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*. DISKI-Reihe Nr. 126, infix, Berlin: Akad. Verl.-Ges. Aka.

Weitere Themen der Vorlesung

Einführung und Hintergrund:

- § 1 Einführung ins fallbasierte Schließen √
- § 2 Kognitionswissenschaftlicher Hintergrund √

Grundtechniken:

- § 3 Fallrepräsentation √
- § 4 Ähnlichkeitsbestimmung √
- § 5 Retrieve: Effiziente Fallauswahl √
- § 6 Reuse: Lösungsanpassung
- § 7 Revise: Lösungsfeedback
- § 8 Retain: Lernen

Fallbasierte Systeme für spezielle Aufgabenkategorien:

- § 9 Fallbasierte Klassifikation, Diagnose & Entscheidungsunterstützung
- § 10 Fallbasierte Konfiguration und Design
- § 11 Fallbasierte Planung