

# 4. Retrieval-Verfahren

**Retrieval = Erinnern**

Bereitstellen von Information: Frühere Fälle

- nützlich für Problemlösung
- alternative Möglichkeit
- akzeptabel als Antwort

⇒ a-posteriori Kriterien, subjektiv

Abschätzen anhand von

- partieller Übereinstimmung,
- ungefähre Übereinstimmung
- vager Beschreibungen

⇒ ersatzweise geeignete Ähnlichkeit/Distanz

Technisch:

Datenbankabfragen

Suchmaschinen

Ziel: Präferenz in der Falldatenbasis

- Suchproblem unter erschwerten Bedingungen:  
Ähnlichkeit in einer Datenbank
- Vergleich mit allen Fällen kaum möglich
- Indizierungstechniken:  
kd-Bäume, CRNs, Nearest-Neighbour, Machine Learning...
- Hierarchische Gliederung der Falldatenbasis:  
Dynamic Memory (Schank, Kolodner)

Jeweils ausgehend von Anfrage  $q$ :

Präferenz-Relation  $>_q$  über Fallbasis  $C$

$$c >_q c' := SIM(q, c) > SIM(q, c')$$

Gewisse Freiheit:

- evtl. nicht alle relevanten Fälle auffinden  
(Unvollständigkeit)
- evtl. auch nicht relevante Fälle liefern  
(Unkorrektheit)

Kritik durch Nutzer erforderlich

**Definition: Recall, Precision:**

(*Vollständigkeit* bzw. *Korrektheit* der Retrievalprozedur)

$$Recall := \frac{\text{Gefundene relevante Fälle}}{\text{Relevante Fälle}}$$

$$Precision := \frac{\text{Gefundene relevante Fälle}}{\text{Gefundene Fälle}}$$

Messung durch Experimente

Probleme:

- Oft nicht klar, welche Fälle “relevant” sind  
Kriterien bzgl. „Relevanz“?
- Redundanzen evtl. störend

Unter theoretischem Gesichtspunkt erwünscht:

- Hohe Werte (“1”) für Recall und Precision
- Relevante Fälle = ähnlichste Fälle

## Lineare Suche

- $SIM(q, c)$  für alle Elemente  $c \in C$  berechnen,
- $C$  gemäß  $>_q$  ordnen,
- beste Fälle ausgeben.

## Voronoi-Diagramme

komplexe Berechnungen bei  $k > 2$   
(Retrieval, Konstruktion)

## kd-Bäume für FBS

(nach S.WESS u.a.)

Jeweils Verzweigung in Unterbäume  
gemäß Test bzgl. eines Merkmals.

Tests nach unterschiedlichen Kriterien erzeugt:

kd-Bäume für Datenbanken:

balancieren bzgl. gleichmäßig großer Unterbäume

kd-Bäume für Machine-Learning mit ID3:

Unterbäume jeweils möglichst homogen bzgl. Klassen  
(Rest-Unsicherheit minimieren)

kd-Bäume für FBS

ähnliche Fälle in gleiche Unterbäume

# Rekursives Erzeugen eines kd-Baumes

Rekursions-Schritt:

Aus aktuell betrachteter Fallmenge

$$C = \{c_1, \dots, c_d\}$$

$$c_i = [c_i(1), \dots, c_i(k)] \in W_1 \times \dots \times W_k$$

und vorgegebener "Bucketgröße"  $b$ :

- Falls  $d = \text{card}(C) \leq b$ :  
Blattknoten, der  $C$  repräsentiert.
- Falls  $\text{card}(C) > b$ :
  - Wähle ein **günstiges** *Diskriminatorattribut*  $i \in \{1, \dots, k\}$  und einen **günstigen** *Partitionswert*  $p \in W_i$ .
  - Partitioniere  $C$  in  $C_{<} := \{c \mid c(i) < p\}$  und  $C_{\geq} := \{c \mid c(i) \geq p\}$
  - Konstruiere rekursiv Unterbäume für  $C_{<}$  und  $C_{\geq}$ .
  - Ergebnis ist ein Baum mit dem Test  $c(i) < p$  in der Wurzel, dem linken Unterbaum für  $C_{<}$  und dem rechten Unterbaum  $C_{\geq}$ .

Damit repräsentiert jeder Unterbaum (zunächst) die bei seiner Konstruktion verwendete Menge  $C_{<}$  bzw.  $C_{\geq}$ .

“Primärer” Retrievalprozeß wie üblich:

Für Anfrage  $q = [q(1), \dots, q(k)]$

wird jeweils beim Test “ $q(i) < p$  ?”

- bei positivem Ergebnis nach links verzweigt,
- bei negativem Ergebnis nach rechts.

(bei unbekanntem Wert: beide Unterbäume weiter verfolgen)

Jeder Unterbaum repräsentiert ein Intervall  
im Merkmalsraum  $W_1 \times \dots \times W_k$ .

Primäres Retrieval für eine Anfrage  $q = [q(1), \dots, q(k)]$   
liefert alle Fälle  $c_i \in C$ , die in dem erreichten Blattknoten (Bucket)  
repräsentiert sind  
(d.h. die aus dem betreffenden Intervall sind).

### **Problem:**

Wenn die Anfrage  $q$  relativ dicht an der Intervallgrenze liegt, werden evtl. ähnliche Fälle aus den Nachbarintervallen beim primären Retrieval nicht gefunden.

## Erweiterung bzgl. Test von Nachbarintervallen

Aufgabe:  $n$  nächste Nachbarn aus  $C$  zu Anfrage  $q$  finden,  
Ähnlichkeit als inverse (Euklid.) Distanz.

Retrieval:

1. Primäres Retrieval für  $q$  durchführen.

Gefundene Fälle (Bucket) gemäß Nähe zu  $q$  sortieren:

$$g_1, \dots, g_n, \dots, g_b$$

(Annahme: Bucketgröße  $b \geq n$ , andernfalls Nachbarintervalle einbeziehen)

$g_1, \dots, g_n$  sind Kandidaten ( $g_n$  ist der "schlechteste").

2. *Ball-Within-Bounds-Test*:

Überprüfen, ob die Kugel mit dem Radius  $Abstand(x, g_n)$  vollständig innerhalb der aktuellen Intervallgrenzen liegt.

Ja: fertig.

Nein: Nachbarintervalle prüfen

(Backtracking im kd-Baum).

3. Für alle betreffenden Intervalle:

*Bounds-Overlap-Ball-Test* (BOB):

Überprüfen, ob die Kugel mit dem Radius  $Abstand(x, g_n)$  teilweise innerhalb der Intervallgrenzen liegt.

4. Sequentiell für alle Intervalle mit positivem BOB-Test nach weiteren ähnlichen Fällen suchen:

Neuberechnung von  $g_1, \dots, g_n$ .

5. Erneuter *Ball-Within-Bounds-Test*: Überprüfen, ob die Kugel mit dem Radius  $Abstand(x, g_n)$  vollständig innerhalb der bisher untersuchten Teilräume liegt.

Weiter für Ja/Nein analog zu 2.

In ungünstigen Fällen muß der gesamte Baum durchsucht werden.

Günstige Konstruktion des kd-Baumes:

Alternativen bzgl.

- Wahl des Diskriminatorattributes:  
statistische Verfahren bezogen auf Ähnlichkeit
- Wahl des Partitionswertes:  
statistische Verfahren  
Maximum-Splitting
- Wahl der Bucketgröße:  
konstant (Außenspeicher)  
Clusterähnlichkeit  
Begriffszugehörigkeit

Globale Optimierung:

Diskriminatorattribut und Partitionswert gemeinsam bestimmen  
(z.B. Maximale durchschnittliche Ähnlichkeit der Fälle in den Partitionierungen)

Weitere Optimierung:

Minimale und maximale dynamische Intervallgrenzen.

Weiter:

Verweise auf Fälle aus Nachbarintervallen.

## kd-Bäume implementieren

### Akkumulation für Zurückweisung :

Einzelne große Distanz  $\rightarrow$  Zurückweisung.

Nächster Nachbar: *alle* lokalen Werte sind nah.

Methode: Entscheidungsbaum

(“top down pruning”)

- Distanz:  
akkumuliert *Argumente für Ausschluß*

Akkumulation für Zurückweisung

*Akkumulation für Akzeptanz*: Case Retrieval Netze  
(im Zusammenhang mit *Fallvervollständigung*, s.u.)

## Hillclimbing auf Fallnachbarschaften

Vorbereitung des Retrievalverfahrens:

Auswahl von Repräsentanten aus Falldatenbasis  $C$ ,

Berechnung aller Nachbarschaftsbeziehungen für Repräsentanten.

Retrievalverfahren (für neuen Fall  $q$ ):

Start bei einem (“günstigen”) Repräsentanten  $r_0$ .

Unter den Nachbarn von  $r_0$  wird der zu  $q$  nächste als  $r_1$  bestimmt.

analog:  $r_2, r_3, \dots$  bis keine Verbesserung mehr möglich.

mögliche Probleme:

Zyklen, lokale Maxima,

weitere ähnliche Fälle in  $C$ .

# 5. Adaption

Übertragung auf ähnliche Fälle:

- Direkte Übertragung:  
⇒ Verallgemeinerung (Lernverfahren)
- Anpassung an aktuelles Problem
  - Reparatur von Fehlern, z.B. falscher Klassifikation
  - Interpolation (stetiger Problemlösungsraum)

Anpassungsformeln bzw. -regeln:

Abweichung der Probleme ⇒ Abweichung der Lösung

- Kombination der Lösungen mehrerer Fälle

Beispiel Preisvergleich: Verkaufspreis "Dein Fahrrad"

Anfrage:

Nr.	Baujahr	Typ	Marke	Gänge	Beleuchtung	Preis
	1987	Damenrad	Diamant	1 / 3	ja	?

Fälle einer Falldatenbasis:

Nr.	Baujahr	Typ	Marke	Gänge	Beleuchtung	Preis
1	1938	Damenrad	Singer	1 / 1	nein	300,-
2	1995	MB	MCB	3 / 6	nein	1000,-
3	1989	Kinderrad	Diamant	1 / 3	ja	100,-
...						

Beispiel Reiseberatung: Reisewunsch

Nr.	Ort	Typ	Zeit	Personen	Kategorie	Preis
	?	Baden	12.-18.2.95	4	Ferienwohnung	?

Fälle einer Falldatenbasis (Angebote):

Nr.	Ort	Typ	Zeit	Personen	Kategorie	Preis
1	Paris	Essen	12.-18.2.95	2	****	3000,-
2	Potsdam	Kultur	20.-25.6.95	2	Zelt	500,-
3	Peking					
...						

Beispiel Fahrrad-Reparatur: Platter Vorderreifen.  
 (Unbekannte Ursache: Nagel im Schlauch)

Ähnliche Fälle einer Falldatenbasis:

Nr.	Problem	Diagnose	Therapie
1	Platter Vorderreifen	Rad nicht aufgepumpt	Pumpen
2	Platter Vorderreifen	Ventil defekt	Ventil wechseln Pumpen
3	Platter Hinterreifen	Poröser Schlauch	Hinterrad ausbauen Schlauch abnehmen Schlauch prüfen Schlauch flicken Pumpen Schlauch prüfen Schlauch montieren Pumpen Hinterrad einbauen
4	...		

Praktische Anwendung:

Lösung aus Fall 1 scheitert.

Lösung aus Fall 2 scheitert.

Adaption Fall 3:

Problem: *Vorderrad* ähnlich *Hinterrad*

Diagnose: *Poröser Schlauch* ähnlich “*platt*”

## Lösungsanpassung

angestrebte Aktion: Ausgebauten Schlauch prüfen.

Dazu: Vorderad (!) ausbauen, Schlauch abnehmen.

Neues Problem: Vorderrad ausbauen

ähnliche Fälle:

Nr.	Problem	Diagnose	Therapie
4	Vorderrad läuft unrund	Vorderradachse gebrochen	Vorderrad ausbauen Achse auswechseln Vorderrad einbauen
5	...		

“Skript” für Vorderrad ausbauen (Fall 4):

Rad umdrehen,  
Muttern lockern,  
Rad aus Gabel nehmen (Bremsen beachten).

Nächstes Problem: Schlauch prüfen

Skript wie unter Fall 3. Ergebnis: Schlauch defekt (Nagel).

Nächstes Problem: Schlauch reparieren, prüfen

Skript aus Fall 3. ...

...

Nächstes Problem Vorderrad einbauen

Skript aus Fall 4. ...

Anpassung durch System:

- z.B. Preis an Menge anpassen
- Kombination aus unterschiedlichen Fällen

Anpassung als interaktive Arbeit:

- Präsentation früherer Fälle
- Auswahl durch Nutzer:  
Fälle geben Hinweise, Argumentationen, ...

Anwendung bestätigt/widerlegt Nützlichkeit

## 6. Aktualisierung des Systems

- Neue Fälle
  - Aufbereiten: Struktur, Indizes
  - Eintragen in Fallsammlung, Indizieren
- Retrievalsystem erweitern
  - neue Indizes
  - Ähnlichkeit erweitern
- Ähnlichkeit anpassen
  - bessere Akzeptanz (Lernverfahren)
- Vergessen
  - Streichen irrelevanter Fälle
  - Retrieval „alter“ Fälle vermindern

Verfahren abhängig von Domäne:

- Katalog (Last-Minute-Reisen): vollautomatisch
- Diagnose-Dokumente
  - Einfügen neuer Begriffe durch Nutzer  
(mit System-Unterstützung)
  - Einfügen neuer Dokumente: vollautomatisch

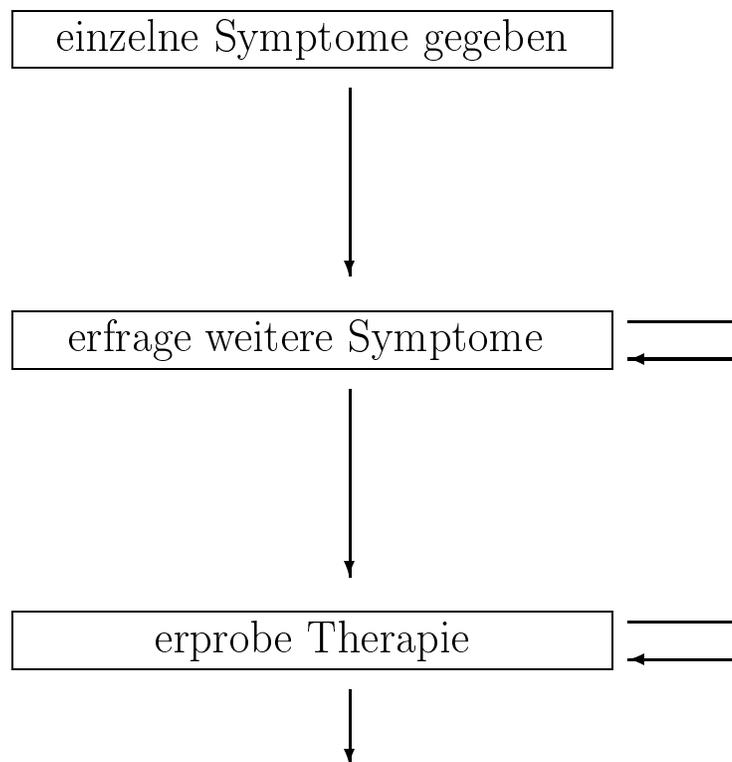
# 7. Erweiterte Betrachtung: Fallvervollständigung

Klassisch:

Fall = Problem + Lösung:  $\boxed{P} + \boxed{L}$

⇒ Regelartig: Wenn Problem, dann Lösung.

Dagegen: Diagnose als *Prozeß*



Aktionen erfolgen in der realen Welt

- abhängig von *Entscheidungen*
- liefern weitere Informationen

interaktive Arbeit Nutzer – FBS-System

Angemessene Betrachtung:

**Problemlösen als Prozeß:** “Completing a task”

**unter Gesichtspunkt der Informationsbearbeitung:**

(neben den eigentlichen “praktischen Schritten”)

- Ausgangspunkt: Unvollständige Information  
(nicht ausreichend für unmittelbare Problemlösung)
- Wiederholte Schritte: Vervollständigen von Informationen  
(bis zu einem befriedigenden Abschluß)

“Completing information”

Beispiele: Diagnose, Design, Beratung, ...

Fall = Gesamtheit der am Ende angesammelten Information

Fallvervollständigung: Prozeß der Informationsammlung

# Fallvervollständigung (Case Completion)

Unterstützung komplexer Aufgaben (Problemlösungsprozesse)

Strukturierung der Fälle in

## **Informationseinheiten (IE)**

(z.B. Attribut-Werte-Paare, Schlüsselwörter, etc.)

Informationssammlung liefert neue IEs für aktuelles Problem:

“Fallvervollständigung”

Wechsel des Gebrauchs von IEs:

- **Kurzschluß** als Lösungsvorschlag (“Ausgabe”)
- **Kurzschluß** als Symptom (“Eingabe”)

## Charakteristika der “Fallvervollständigung”

- Sammlung von Information  
in der Auseinandersetzung mit der **realen** (!) Welt:
  - eigene Entscheidungen
  - Resultate
- Unterstützung durch CBR:
  - Anleitung/Hilfestellung für  
Auswahl von Aktionen im Problemlösungsprozeß  
(sinnvolle Aktionen, mögliche Resultate etc.)

Fallvervollständigung ist eine allgemeine Sicht auf FBS-unterstützte Problemlösungsprozesse. Die Bearbeitung eines neuen Falls beginnt mit unvollständiger und vager Information. Während des Problemlösungsprozesses werden beim Agieren in der realen Welt weitere Informationen gesammelt. FBS kann Argumente für anstehende Entscheidungen liefern durch den Vergleich mit vollständigeren Fällen aus der Vergangenheit. Abgeschlossene Fälle können als neue Erfahrung zur späteren Verwendung gespeichert werden.

Statt Regelform („Wenn Problem, dann Lösung“)

Fälle (Episoden, „Erfahrungstatsachen“)  
in der Form von **Constraints**.

Beobachtung: Bestimmte Bedingungen gelten zusammen.

Annahme (FBS): Diese Bedingungen gehören zusammen.

Verallgemeinerung (ML): Diese Bedingungen gelten immer zusammen.

Abfrage analog zu Datenbanken bzw. Prolog:

Einzelne Fakten eines Falles gegeben

- welche Fakten können dazu gehören.

Probleme als „unvollständige“ Beschreibungen von Fällen.

Gemeinsame Form: Anfragen und Fälle sind

*Mengen von Informationseinheiten (IE)*

$$q = q_1, \dots, q_k$$

$$c = c_1, \dots, c_n$$

- Ähnlichkeit zwischen Anfrage  $q$  und Fall  $c$   
bei Ähnlichkeit zwischen IEs  $q_i$  und  $c_j$ .
- Prinzipiell: Vergleich aller  $q_i$  und  $c_j$  (z.B. bei Texten).
- Bei strukturierten Fällen (z.B. IEs als Attribut-Werte-Paare):  
Nur IEs vom gleichen Typ (gleiches Attribut) vergleichen.  
( $\Rightarrow$  Kompositorische Ähnlichkeitsfunktion)

Anforderung an Retrieval für Fallvervollständigung:

- Umgang mit unvollständigen Beschreibungen

⇒ Akkumulation für Akzeptanz

### IDEE: CASE RETRIEVAL NETZE

- Anfragen/Fälle bestehen aus Informationseinheiten (IE)  
IE z.B. **Reiseziel=Santorini**
- Propagierung von Ähnlichkeiten zwischen IE  
z.B. **Reiseziel=Santorini**  
ist ähnlich zu **Reiseziel=Rhodos**
- Propagierung von IE zu Fällen  
z.B. **Reiseziel=Rhodos** gehört zu Angebot 28461

# 8. Basic Case Retrieval Net (BCRN)

## Definition

*Ein Basic Case Retrieval Net (BCRN) ist definiert durch*

*$N = [E, C, \sigma, \rho, \Pi]$  mit*

*$E$ : endliche Menge von Informations-Einheiten (IE-Knoten),*

*$C$ : endliche Menge von Fall-Deskriptoren (Fall-Knoten),*

*$\sigma$ : Ähnlichkeitsfunktion*

$$\sigma : E \times E \rightarrow \mathcal{R}$$

*zur Angabe von Ähnlichkeiten  $\sigma(e', e)$  zwischen IE  $e'$ ,  $e$ ,*

*$\rho$ : Relevanz-Funktion*

$$\rho : E \times C \rightarrow \mathcal{R}$$

*zur Angabe der Relevanz  $\rho(e, c)$  von  $e$  für Retrieval von  $c$ .*

*$\Pi$ : Menge von Propagierungs-Funktionen  $\pi_n$  für jeden Knoten  $n \in E \cup C$  mit*

$$\pi_n : \mathcal{R}^E \rightarrow \mathcal{R}.$$

Weitere Angaben z.B.:

- Die Funktionen  $\pi_n$  sind monoton.

„Zustand“ eines Netzes:

## Definition

Die Aktivierung eines BCRN  $N = [E, C, \sigma, \rho, \Pi]$  ist eine Funktion

$$\alpha : E \cup C \rightarrow \mathcal{R}.$$

„Standard Form“ einer Anfrage:

$$\alpha_{query}(e) = \begin{cases} 1 & : \text{für IE Knoten } e \text{ des neuen Problems} \\ 0 & : \text{sonst} \end{cases}$$

Speziellere Anfragen:

$\alpha_{query}$  mit speziellen („Wichtigkeits“-)Werten für spezielle IE.

„Kontext“, „Fokus“: weitere initiale Aktivierungen.

## Propagierung:

### Definition

*Gegeben: BCRN  $N = [E, C, \sigma, \rho, \Pi]$  with  $E = \{e_1, \dots, e_s\}$ .*

*$\alpha_t : E \rightarrow \mathcal{R}$  sei Aktivierung zur Zeit  $t$ .*

*Die Aktivierung von IE-Knoten  $e \in E$  zur Zeit  $t+1$  ist gegeben durch*

$$\alpha_{t+1}(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), \dots, \sigma(e_s, e) \cdot \alpha_t(e_s)),$$

*und die Aktivierung von Fall-Knoten  $c \in C$  zur Zeit  $t+1$  ist gegeben durch*

$$\alpha_{t+1}(c) = \pi_c(\rho(e_1, c) \cdot \alpha_t(e_1), \dots, \rho(e_s, c) \cdot \alpha_t(e_s)).$$

### Restriktion auf 2 Schritte:

#### **Schritt 1 :**

Die Anfrage-Aktivierung  $\alpha_{query}$  wird propagiert zu allen IE-Knoten  $e \in E$ :

$$\alpha_1(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_{query}(e_1), \dots, \sigma(e_s, e) \cdot \alpha_{query}(e_s)),$$

#### **Schritt 2 :**

Das Resultat von Schritt 1 wird propagiert zu den Fall-Knoten  $c \in C$ :

$$\alpha_2(c) = \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), \dots, \rho(e_s, c) \cdot \alpha_1(e_s)).$$

## Resultat

Gegeben: BCRN  $N = [E, C, \sigma, \rho, \Pi]$  mit  $E = \{e_1, \dots, e_s\}$ .

Die resultierende Aktivierung  $\alpha_N : C \rightarrow \mathcal{R}$

von Fall-Knoten bezüglich einer Anfrage  $\alpha_{query} : E \rightarrow \mathcal{R}$

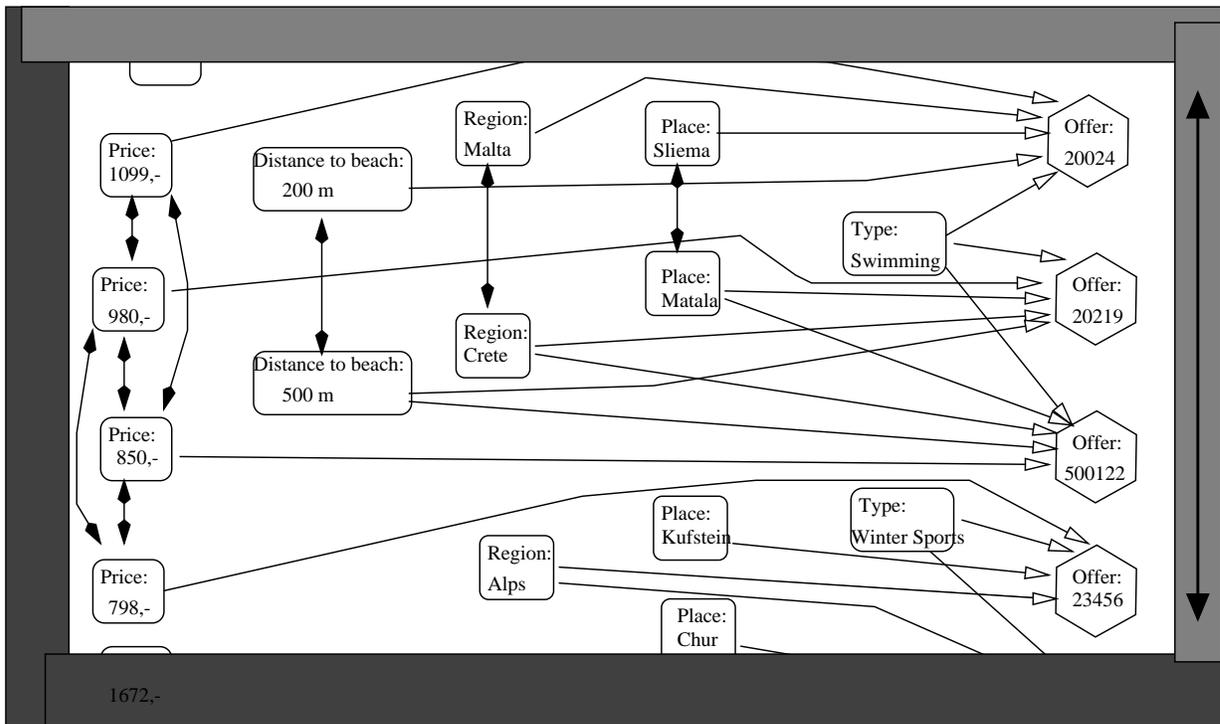
ist gegeben (nach 2 Schritten) durch

$$\begin{aligned} \alpha_N(c) = & \pi_c( \rho(e_1, c) \cdot ( \pi_{e_1}( \sigma(e_1, e_1) \cdot \alpha_{query}(e_1), \dots \\ & \dots, \sigma(e_s, e_1) \cdot \alpha_{query}(e_s) ), \\ & , \dots, \\ & \rho(e_s, c) \cdot ( \pi_{e_s}( \sigma(e_1, e_s) \cdot \alpha_{query}(e_1), , \dots \\ & \dots, \sigma(e_s, e_s) \cdot \alpha_{query}(e_s) ) ) ). \end{aligned}$$

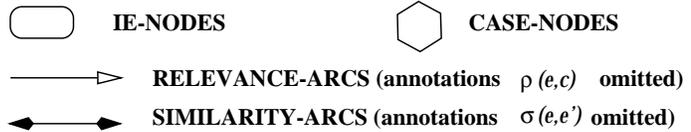
Das Resultat des Fall-Retrievals bzgl. der Anfrage  $\alpha_{query}$

ist die Präferenz-Ordnung  $\succ_N$  der Fälle gemäß

abnehmender Aktivierungen  $\alpha_N(c)$  von Fall-Knoten  $c \in C$ .



**PART OF A CASE RETRIEVAL NET:**  
(Travel Agency Domain)



**Fall:** Reise-Angebot.

Beschrieben durch **Informations-Einheiten (IE)**.

Beispiel:

Falldeskriptor: <Offer 20219>.

IEs: <Type:Swimming>,  
<Price:980,->,  
<Place:Matala>,  
<Region:Crete>,  
<Distanz to beach:500 m>

## **Anfrage:**

*Reise nach Kreta zum Schwimmen in Strandnähe.*

## **Initiale Aktivierung**

IE-Knoten

<Type:Swimming>, <Distanz to beach:200 m>, <Region:Crete>.

## **Aktivierungsausbreitung gemäß IE-Ähnlichkeit:**

IE Knoten

<Region:Malta>, <Distanz to beach:500 m>.

## **Aktivierungsausbreitung gemäß Relevanz:**

Angebote

<Offer 20024>, <Offer 20219>, <Offer 500122>.

## **Resultat:**

Liste meist aktivierter Angebote.

*(Präferenz-Ordnung der Fälle)*

## Problem:

Großer Wertebereich  $W$  für ein Attribut  $A$

(z.B. Preisangaben: nicht alle Werte als IEs)

Annahmen:

1. Gegeben: Attribut  $A$  mit Wertebereich  $W$ ,  
entsprechende IEs sind Attribut-Werte-Paare  $e_{(A,w)}$

$$E_A := \{e_{(A,w)} \in E \mid w \in W\} \subseteq E$$

ist Menge aller IE bzgl. Attribut  $A$ .

2. IE bzgl. eines Attribut  $A$  können nur zu anderen IE bzgl. des gleichen Attribut ähnlich sein, d.h.:

$$e \in E_A \wedge e' \notin E_A \rightarrow \sigma(e, e') = 0 \wedge \sigma(e', e) = 0$$

3. Falls ein Attribut-Wert-Paar  $(A, w)$  Bestandteil einer Fallbeschreibung ist, so existiert ein IE-Knoten  $e_{(A,w)} \in E$  für  $(A, w)$  und eine verbindende Kante zum entsprechenden Fall-Knoten. (Andernfalls: Kein IE für  $(A, w)$  vorhanden.)

4. Es existiert eine *algorithmische Prozedur* zur Berechnung der Merkmals-Ähnlichkeit  $\sigma_A : W \times W \rightarrow \mathcal{R}$  für das Attribut  $A$  so daß gilt:

$$\sigma(e_{(A,w)}, e_{(A,w')}) = \sigma_A(w, w').$$

Für IE-Knoten  $e = e_{(A,w')} \in E$  ist Beitrag von  $e_j = e_{(A,w)}$  in

$$\alpha_{t+1}(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), \dots, \sigma(e_s, e) \cdot \alpha_t(e_s)),$$

gegeben durch

$$\sigma(e_j, e) \cdot \alpha_t(e_j) = \sigma(e_{(A,w)}, e_{(A,w')}) \cdot \alpha_t(e_{(A,w)}) = \sigma_A(w, w') \cdot \alpha_t(e_{(A,w)})$$

Problem: Anfrage kann sich auf Wert  $w$  für  $A$  beziehen, wobei kein IE-Knoten  $e_{(A,w)}$  in  $E_A$  existiert.

### **Simulation fehlender IE-Knoten $e_{(A,w)}$ für Attribut $A$ durch „Berechnungsknoten“ $n_A$ :**

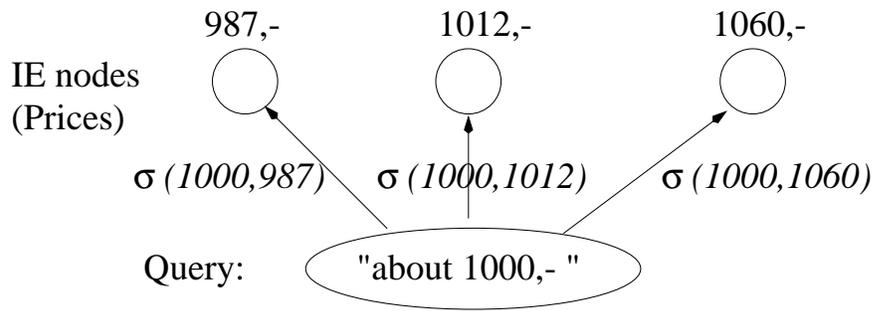
Berechnung der Ähnlichkeiten:

- $n_A$  verbunden mit allen IE-Knoten  $e_{(A,w')} \in E_A$  durch Bögen mit variablem Gewicht  $\sigma_A(w, w')$ .  
Gewicht berechnet und zugewiesen durch  $n_A$ .

alternativ:

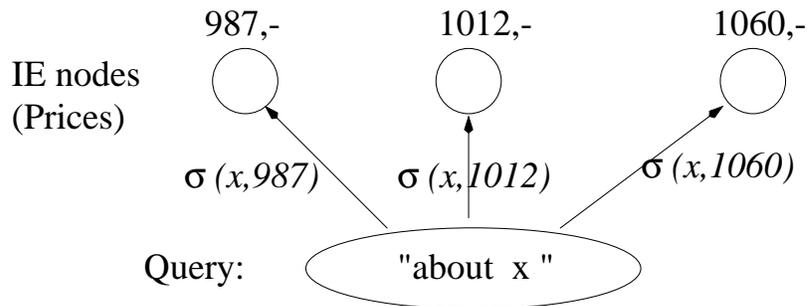
- $n_A$  verbunden mit allen IE-Knoten  $e_{(A,w')} \in E_A$  durch Bögen mit variablem Gewicht  $w$ .  
Berechnung der Aktivierung in Propagierungsfunktion  $\pi$ .

Beispiel für Implementation von fehlenden Werten:



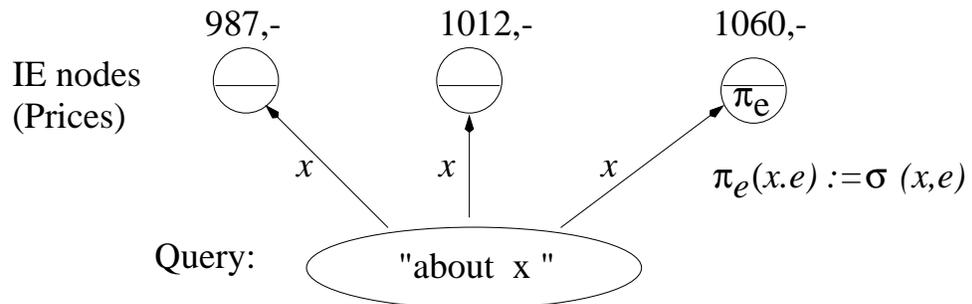
Intended similarity

---



Intended similarity (general case)

---



Implementation with propagation functions in IE nodes

---

# Kompositorische Ähnlichkeitsfunktionen in BCRN

## Satz

*Jede kompositorische Ähnlichkeitsfunktion  $\sigma$  kann durch ein BCRN  $N = [E, C, \sigma, \rho, \Pi]$  berechnet werden:*

*Sei  $c_{[c_1, \dots, c_n]} \in C$  der Falldeskriptor*

*für den Fall  $[c_1, \dots, c_n]$ ,*

*und sei  $[q_1, \dots, q_n]$  eine Anfrage.*

*Wenn  $[q_1, \dots, q_n]$  in die Anfrageaktivierung*

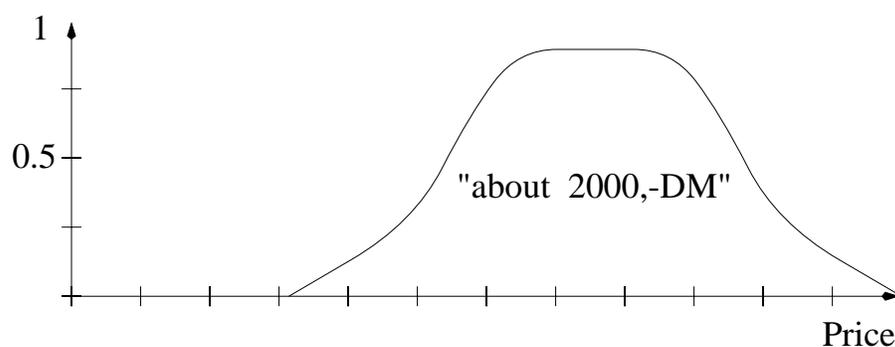
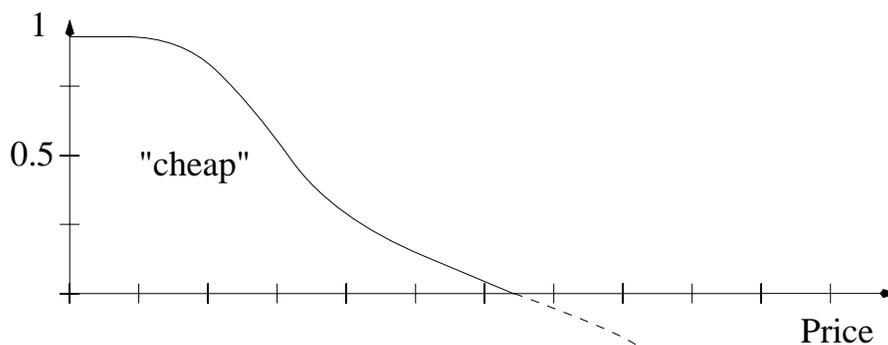
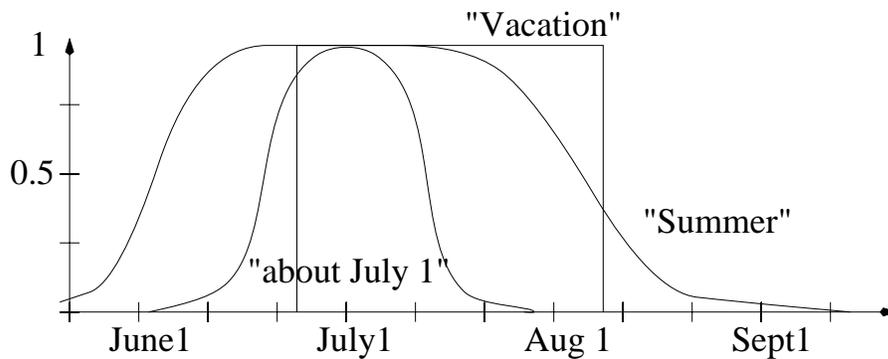
$$\alpha_{query}(e) = \begin{cases} 1 & : \text{für } e = [A_i, q_i] \\ 0 & : \text{else} \end{cases}$$

*transformiert wird, so berechnet das BCRN:*

$$\alpha_N(c_{[c_1, \dots, c_n]}) = \sigma([c_1, \dots, c_n], [q_1, \dots, q_n]).$$

## Lokale Ähnlichkeiten (Vage/fuzzy Begriffe):

- Linguistische Terme
- Vage Beschreibung konkreter Werte („ungefähr 2000 DM“)



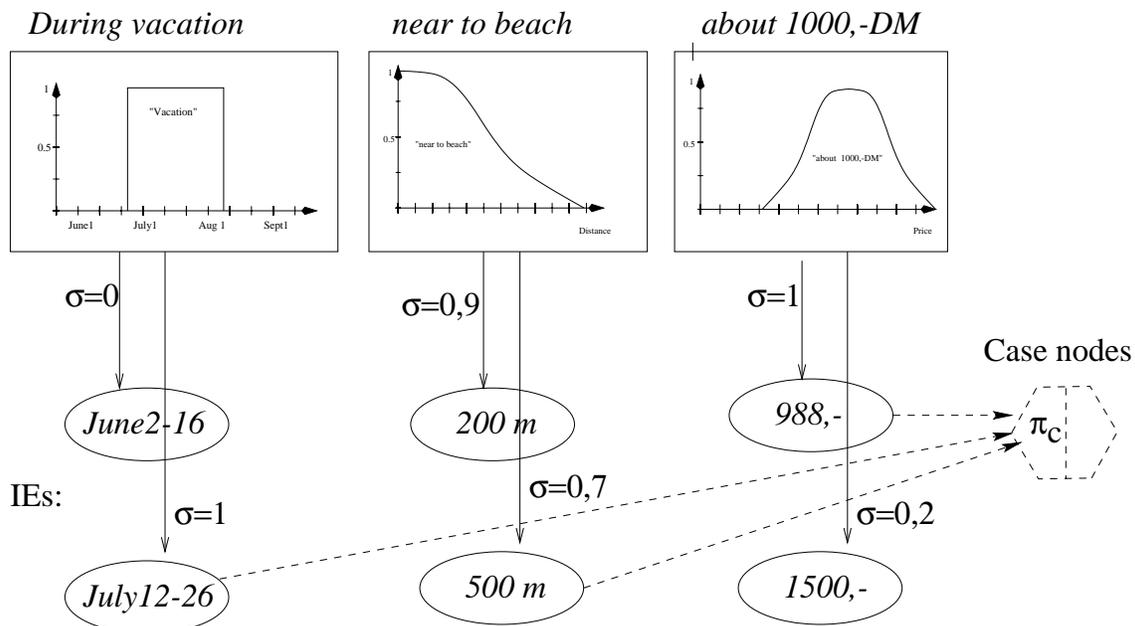
Interpretation von Fuzzy-Mengen (Charakteristischer Fkt.) für das Retrieval („**erinnern**“):

“Wichtigkeit beim Wert  $x$  an den linguistischen Term zu denken”  
(Negative Werte: Nicht erinnern)

# Komposition:

Beispiel:

Query:



Berechnung am IE-Knoten  $e$ :

$$\sigma(e_q, e) = f_e(e_q)$$

mit charakteristischer Funktion  $f_e$  für linguistischen Term  $e$

Akkumulation am Fall-Knoten  $c$  durch Propagierungsfunktion  $\pi_c$   
(z.B.  $\Sigma$ , Max, Min,  $\dots$ , – Normalization ?)

→ Ranking der Fälle bzgl. Anfrage

Kombination der eingehenden Aktivierungen an den Knoten z.B. mittels Summe:

$$\alpha(e) = \sum_{e'} \sigma(e', e) \cdot \alpha_{query}(e')$$

$$\alpha(c) = \sum_e \rho(e, c) \cdot \alpha(e)$$

Ergebnis:

Gewichtete Summe mit Fall-spezifischen (!) Gewichten

$$\alpha(c) = \sum_e \rho(e, c) \cdot \sum_{e'} \sigma(e', e) \cdot \alpha_{query}(e').$$

Häufig besser:

Maximum statt Summe für  $\pi_e$  (z.B. im Text-Retrieval)

## Spezielle Möglichkeiten/Eigenschaften:

- modulare Struktur
- flexibel
- leichte Veränderbarkeit
- unterschiedliche Arten von IE
- Ähnlichkeit zwischen beliebigen IE
- kompositorische Ähnlichkeitsfunktionen
- Ähnlichkeitsfunktionen nicht notwendig symmetrisch
- individuelle Ähnlichkeitsfunktionen für jedes  $c \in C$ .
- Variationen in der Anfrage-Aktivierung  $\alpha_{query}$
- „bottom-up“ Fall-Rekonstruktion (Fallvervollständigung) statt „top-down“ Ausschlußverfahren
- Fähigkeit zu „Kompromissen“
- Effektive Retrieval-Prozedur
- Fehlende Werte unproblematisch
- „Evidenz-Werte“ für die Fälle
- Parallele Arbeit
- Brücke zu subsymbolischen Verfahren
- Visualisierung

## Verwandte Modelle

- Assoziative Speicher,
- Semantische Netze,
- Conceptual Dependency/Scripts/Dynamic Memory,
- Discrimination Networks,
- Neuronale Netze
  - rekurrent
  - strukturiert

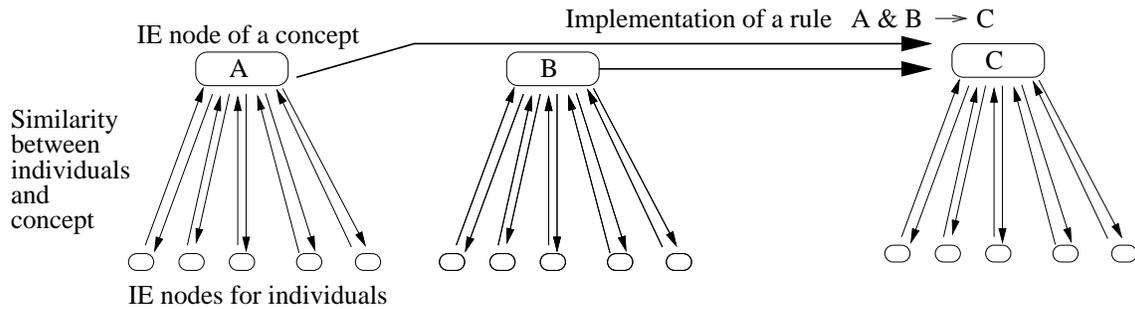
Offenes Problem:

Lernverfahren von Neuronalen Netzen für Ähnlichkeiten

## Erweiterungen

- Aktivierungs-Ausbreitung über mehrere Takte  
⇒ Spreading Activation
  - lazy evaluation
  - “any-time” retrieval
- Ausbau für Fall-Vervollständigung  
(effiziente Tests vorschlagen)
- Einbindung von „Konzepten“  
(strukturelle/konzeptionelle Ähnlichkeiten)
- Verbindungen für has-part, is-a, ... -Hierarchien
- micro features
- Kombinierte Konzepte
  - Fall als „horizontale“ Vereinigung
  - Generalisierungs als „vertikale“ Vereinigung
- Regeln, Constraints
- Subsymbolische Berechnungen

## Beispiel: **Konzepte, Regeln**



Konzept "*Karibik*" spart Speicher (für Kanten):

$n^2$  Ähnlichkeitskanten zwischen  $n$  karibischen Orten

ersetzt durch

$2n$  Kanten zwischen Orten und Konzept

Konzepte: **hierarchische** Konstrukte für IEs des gleichen Typs

Fälle: **horizontale** Konstrukte für IEs unterschiedlichen Typs

Regeln verbinden Konzepte:

*Sommer in der Karibik gehört zur Regenzeit.*

Regeln: **horizontale** Konstrukte auf höherer Ebene.

## Spreading activation Nets (SAN)

BCRN zunächst:

Ausbreitung von Aktivierungen in 2 Schritten:

1. gemäß Ähnlichkeit ( $\sigma$ ) zwischen IE-Knoten
2. gemäß Relevanz ( $\rho$ ) von IE-Knoten zu Fall-Knoten

CRN allgemein:

	$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$\dots$
IE-Knoten:	$\alpha_0 = \alpha_{query}$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\dots$
Fall-Knoten:		$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\dots$

Berechnungen über mehrere Takte:

- „any-time-computation“ :  
Fortführung des Retrieval-Prozesses über mehrere Takte bis zu „befriedigender“ Lösung
- weitere Lösung für „Problem der großen Wertebereiche“
- Einbindung von Konzepten usw.

Problematik:

- Zyklen (feedback loops)
- Stabilität, „Abklingen“

→ Simulation von SAN durch BCRN?

→ Compilation in schnellere Netze (Platz vs. Zeit ? )

→ Einfügen „direkter Verbindungen“ (Hebbsche Regel)

## **CRN als kognitives Modell**

Netz als „potentiell Wissen“

( $\Rightarrow$  Scripts, Erinnern = Rekonstruieren)

(Äußere) Impulse führen zu Reaktionen

Informationsfluß über „neuronale Verbindungen“

- direkte unbewußte Reaktion
- direkte bewußte Reaktion
- bewußtes Abwägen vor dem Reagieren

## **Parallele Arbeit**

- Konflikt bzgl. Ressourcen (z.B. Arme, Beine)
- Konflikt bzgl. bewußter Verarbeitung
- Konflikt bzgl. sprachlicher Formulierung

## **Wettbewerb**

- um Ressourcen
- um Aufmerksamkeit

## **Interne Verarbeitung zur Problemlösung**

als Ausbreitung von konkurrierenden Aktivierungen:

*Entscheidung* zugunsten höherer Aktivierung

*Abbruch/Ende* bei Erreichen von „Befriedigung“

**Lernen** durch neue/verstärkte Verbindungen

„Kurzschlüsse“ (Hebbsches Lernen)

Abstraktion: Regelbildung

Verbindung von *reaktivem* und *reflexivem/deliberativem* Handeln

Verbindung von *bewußter* und *unbewußter* Verarbeitung

Verbindung von *symbolischen* und *subsymbolischen* Zugängen

*Erinnern als zentrales Problem*

# CRN und Neuronale Netze

## Strukturelle Verwandtschaft:

- Knoten
- gewichtete Verbindungen
- Aktivierung als „Zustand“

## CRN: Symbolischer Zugang

- keine Trennung zwischen Training und Lernphase
- leichteres Umlernen
- strengere Klassifikation
- weniger fehlertolerant
- bewußte Strukturierung
- Ähnlichkeit muß/kann explizit festgelegt werden (evtl. mit Neuronalen Netzen adaptieren)

## Dynamik:

- Ausbreitung von Aktivierung (Propagierung)
- $\pi$  kann allgemeiner sein als *net*  
technisch einfachere Realisierung,  
z.B. für logische Verknüpfungen
- geeignete Verfahren bzgl. feed-back?
  - abnehmende Aktivierungen/Propagierungen
  - zurücksetzen (nach  $n$  Schritten)
  - einfrieren

## **CRN als paralleles Inferenzmodell:**

(vs. NN als *Muster-Verarbeitung*)

- Auswahl, Präferenz-Entscheidungen
- Mehrstufig arbeiten  
(„Any-time-Computation“)  
(„Argumente sammeln“)
- Fokussieren (Argumente sammeln)
- Unterschiedliche (steuerbare) Propagierungsstrategien
  - Fälle aktivieren
  - Fälle vervollständigen
  - Konzepthierarchien
  - generalisiertes Wissen, Regeln

## Lernen in CRN

mit verschiedenen *Zielen*

- neues Wissen
  - von außen erworben
  - intern erarbeitet
- effektivere Verarbeitung („Compilation“)  
(Fähigkeiten trainieren)
- Anpassung an Probleme
  - „Ähnlichkeiten lernen“

in unterschiedlichen *Formen*:

- neue Fälle („konkret“)
- neue Modelle, Konzepte, Regeln („abstrakt“)
- neue Fähigkeiten (z.B. bzgl. Fallauswahl)

mit verschiedener *Realisierung*

- neue Fall-Knoten
- neue IE-Knoten
- neue Verbindungen
- veränderte Gewichte
- neue Verbindungstypen
- Ersetzung von Verbindungen durch andere (kürzere)  
„Kurzschlüsse“
- Einfügen von Konzepten, Regeln

⇒ *Erfahrungen von ML und speziell NN nutzen*

## Weitere (z.T. offene) Fragen zu CRN

- Implementation
  - Leistungsfähigkeit (Zeit, Umfang)
- Kognitive Modellierung
- Theorie
  - Modellbildung
  - Gegenseitige Simulation, Kompilation
  - Anpassung
  - Lernen
  - negative Werte,  
an unterschiedlichen (evtl. mehreren) Stellen,  
Kombination negativer Werte?
- Anwendungen
- Handhabung
  - Transparenz
  - Visualisierung
  - Programmierung
  - On-line-Anpassung (durch Nutzer)

# CRN für „flaches“ NLP: Retrieval von textuellen Dokumenten

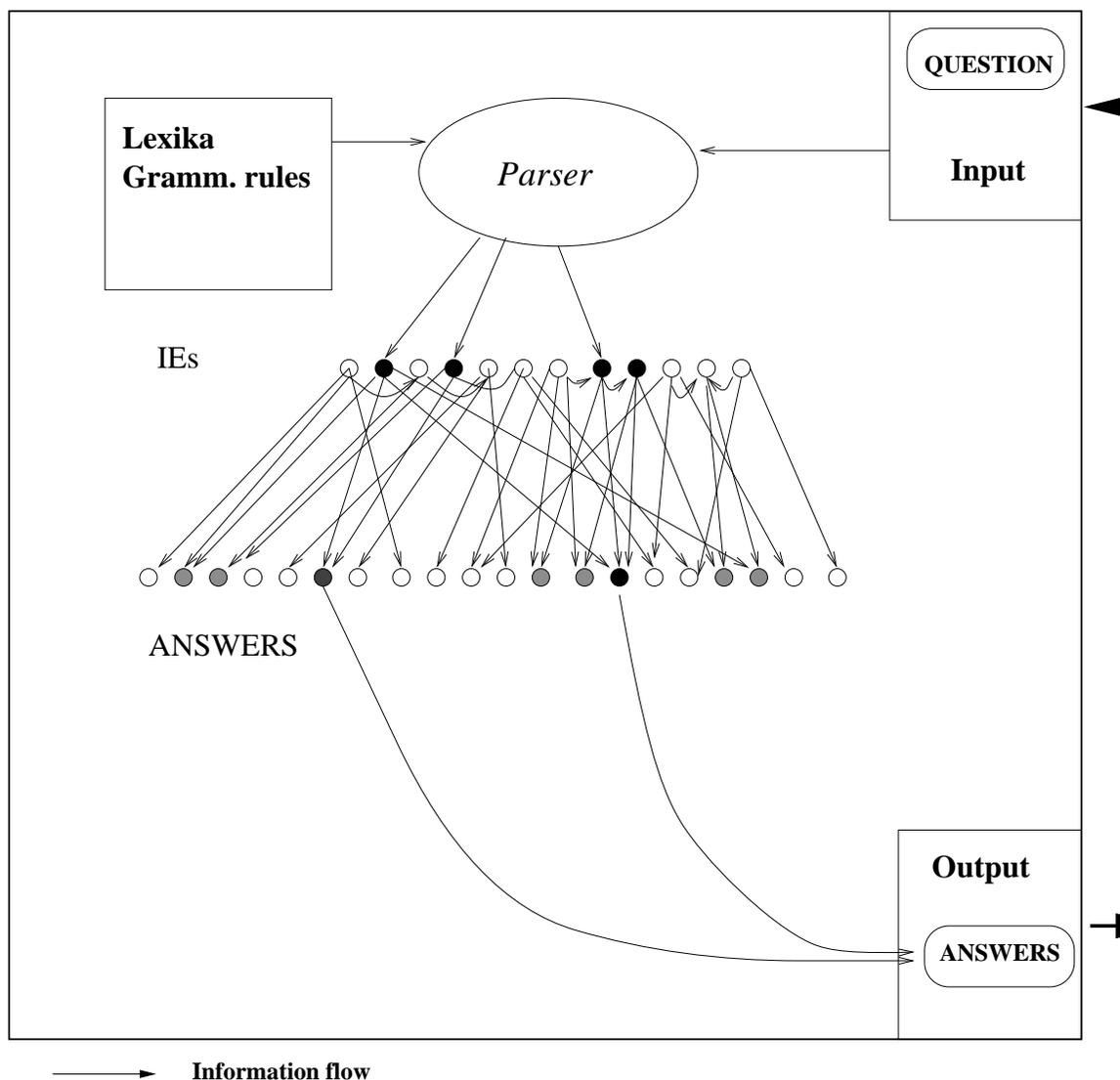
(vgl. Information Retrieval)

Idee:

IEs sind Konzepte

Wörter/Wortgruppen sind „Erscheinungsformen“ der Konzepte  
(Grammatische Formen, Synonyme, andere Sprache, ...)

Ähnlichkeit zwischen Konzepten: Ähnlichkeit von IEs



## Weitere Anwendungen (offene Fragen:

- Steuerung von Agenten:  
Implementation von **thought**
- CRN für Lokalisierung aus unsicheren Sensor-Daten  
IEs als Landmarken  
Fälle = Positionen

## Behandlung “fehlender Werte” in CRNs

= Auslassen von „Argumenten“ für Akzeptanz

In vielen CRNs (z.B. Summierung):  
gleiche Auswirkung wie  $\sigma(e, e') = 0$

Beispiel: Attribut-basierte Anfragen/Fälle:

- In einer Anfrage  $q = [q_1, \dots, q_n]$ :  
Fehlender Wert  $q_i$  benachteiligt Fälle  $c = [c_1, \dots, c_n]$   
mit  $\sigma(q_i, c_i) > 0$
- In einem Fall  $c = [c_1, \dots, c_n]$   
Fehlender Wert  $c_i$  benachteiligt  $c$  bei Anfragen  $q = [q_1, \dots, q_n]$   
mit  $\sigma(q_i, c_i) > 0$

Analog z.B. bei Text-Retrieval.

Anders bei Normierung über vorhandene Werte  
(z.B. analog zu “Simple Matching Koeffizient”)

## 9. Weitere Ansätze

Weitere Ähnlichkeits-Formen:

**Oberflächen-Ähnlichkeit** : (äußerliche) Merkmale.

- Merkmalsvektoren
- Schlüssel-Wörter

**Konzeptuelle (begriffliche, ontologische) Ähnlichkeit** :

- Verwandtschaft innerhalb begrifflicher Systeme bezogen auf
  - Merkmale (z.B. Pflanzensysteme)
  - Funktionalität (z.B. Werkzeuge)
  - Zusammensetzung (z.B. Maschine)
  - is-a, has-part, ...
- Dynamische Konzeptbildung:  
*Dynamic Memory* (SCHANK):  
Memory Organization Packet (MOP)
- Generische Fälle, Abstrakte Fälle

**Strukturelle Ähnlichkeit** ;

- Strukturen, Relationen zwischen Merkmalen
  - grammatikalisch
  - geometrisch

**Erklärungsbasierte Ähnlichkeit** :

- Beschreibungen liefern die Indizes.

## Structure Mapping (GENTNER)

Analogien basieren auf gemeinsamen Strukturen

Analogie/Ähnlichkeit durch **Übertragung von Strukturen**

Dimensionen:

- gemeinsame Attribute (von Objekten)
- gemeinsame Relationen (zwischen Objekten)

Analogie: *Wasser ist analog zu Wärme.*

Literale Ähnlichkeit: *Milch ist wie Wasser.*

Abstraktion: *Wärme ist ein Vektorfeld.*

Anomalie: *Kaffee ist wie das Sonnensystem.*

Erscheinungsform *Die Tischfläche schimmert wie Wasser.*

## Wolke analog zu Schwamm

oberflächlich: rund, weich

relational: zeitweise Wasser speichern

Experiment "Zugriff beim Menschen":

Original (Base)	Analogie	Erscheinung	Falsche Anal.
Geschichte 1		x	
Geschichte 2	x		
Geschichte 3	x		
...			
...			
...			
Geschichte 18			x
Summe: 18 ( + 14 weitere)	6	6	6

1. Aufgabe:

Original erinnern + wiedergeben

2. Aufgabe:

Übereinstimmung zum Original bewerten, max. 5

Bewerten:	4,4	2,8	2,0
-----------	-----	-----	-----

Erkennen:	44%	78%	25%
-----------	-----	-----	-----

# Dynamic Memory - Ansatz nach SCHANK, KOLODNER

Versuche, menschliches Gedächtnis nachzubilden  
“Dynamic Memory”: Gedächtnis entwickelt sich

Gedächtnis als Einheit bzgl.

**Erinnern, Verstehen, Erfahren, Lernen**

Reorganisation als Ergebnis des Verstehens

The structures in memory that are used for processing  
(i.e., the ones that provide expectations and suggest inferences)  
are the same ones that are used for storage

Modelle für Strukturierung

- Conceptual Dependency Theory
- Skripts
- Memory Organization Packets (MOP; “episodic”: E-MOP)
- Thematic Organization Packets (TOP)

## Gedächtnisprinzipien:

1. Erinnern ist beim Menschen oftmals eher ein Vorgang der Rekonstruktion dessen, was hätte geschehen müssen, anstelle eines unmittelbaren Abrufs des tatsächlichen Geschehens.
2. Erinnern erfordert die fortschreitende Annäherung an eine Beschreibung des gesuchten Ereignisses.
3. In einem rekonstruierenden Gedächtnis sind Erinnerungen nicht unmittelbar aufgezählt. Stattdessen müssen die Merkmale, die ein Ereignis beschreiben, rekonstruiert werden.
4. Ähnliche Daten werden im Gedächtnis von einer Kategorie, welche ihrer Übereinstimmung entspricht, anhand ihrer Unterschiede verwaltet.
5. Abrufen aus dem Gedächtnis erfordert Wissen über Kontexte, die mit dem Zielobjekt in Verbindung stehen.
6. Abrufen aus dem Gedächtnis bedarf oft der Suche nach etwas anderem, als eigentlich verlangt war.

Modellbildungen:

**Scripts :**

Komplette (monolithische) Beschreibung einer typischen Ereignisfolge  
(“Restaurant-Besuch”)

**Memory Organization Packets (MOP, E-MOP) :**

Hierarchische Strukturierung von Ereignisfolgen.  
Komposition von Szenen (“Essen”, “Bezahlen”).  
MOP’s als Speichereinheiten auf unterschiedlichen Ebenen.

**Dynamischer Speicher:** Erzeugen, aufspalten, ... von MOP’s.

**Thematic Organization Packets (TOP) :**

Situationen mit verwandten Absichten – analog MOP’s.

Speichern der allgemeinen Information

Unterstützen der Suche nach untergeordneten Spezialisierungen

# Transformational/Derivational Analogy (CARBONELL)

Formalismus im Sinne von GPS

(= General Problem Solver, NEWELL, SIMON, 1963):

Problembeschreibung im Prinzip als

$$\boxed{\text{Anfangszustand } z_0 \Rightarrow \text{Zielzustand } z_f}$$

Operatoren  $o$  zur Zustandstransformation  $z' \Rightarrow^o z''$

Differenzberechnung zwischen Zuständen  $d(z', z'')$

(insbesondere Abstand des aktuellen Zustandes zu  $z_f$ )

Gesucht ist eine Operatorfolge mit

$$z_o \Rightarrow^{o_1} z_1 \Rightarrow^{o_2} z_2 \dots \Rightarrow^{o_n} z_f$$

*Means-ends-analysis (MEA)*:

1. Auswahl eines Operators  $o_{i+1}$  für den aktuellen Zustand  $z_i$  mit

$$z_i \Rightarrow^{o_{i+1}} z_{i+1} \quad \text{und} \quad d(z_i, z_f) > d(z_{i+1}, z_f)$$

2. • Falls  $o_{i+1}$  in  $z_i$  anwendbar ("Vorbedingungen erfüllt"):

Operator anwenden, weiter bei 1.

• Andernfalls:

Neues Teilziel:

Zustand  $z'$  erreichen, der Vorbedingungen für  $o_{i+1}$  erfüllt.

Rekursive Anwendung von MEA zum Erreichen von  $z'$ .

Danach weiter mit 2.

Suchprozeß mit Heuristik gemäß Differenzberechnung

## Transformational Analogy

Aufbauend auf MEA.

Abspeichern von erfolgreichen Plänen (Operatorfolgen).

Für neues Problem  $\boxed{z_0 \Rightarrow z_f}$ :

- Retrieval eines ähnlichen Problems  $\boxed{z'_0 \Rightarrow z'_f}$
- Übertragung (Transformation) der verwendeten Operatorfolge

Ähnlichkeit über Differenzberechnung  $d(z_0, z'_0)$  und  $d(z_f, z'_f)$ .

Transformation der alten Operatorfolge mittels MEA.

## **Derivational Analogy:**

Rekonstruktion/Modifikation relevanter Lösungs -**Verfahren**  
(statt der Lösung wie unter Transformational Analogy).

Alte Fälle beschrieben durch Lösungsschritte jeweils mit

- Problemstellung (Anfangs-, Zielzustand)
- Lösung (Operatorfolge)
- Getroffene Entscheidungen mit Gründen:
  - Auswahl eines Operators
  - Fehlschlag einer alternativen Lösung
  - weitere Alternativen
  - Verwendung von Wissen/Annahmen usw.

Neue Lösung rekursiv konstruieren.

Konstruktion angelehnt an früheres Vorgehen.

Ähnlichkeit (Relevanz) früherer Verfahren:

Vergleichbare Anfangsschritte.

Anpassung:

Übernahme bzw. Korrektur früherer Entscheidungen  
ggf. auch anderweitige Herleitung (z.B. mittels MEA).

Speicherung z.B. mittels MOP's.

# Beispiel: Planungssystem PRODIGY/ANALOGY (VELOSO)

Basis-Planer: NOLIMIT (MEA-Prinzip)

- Zustand: Liste geltender Bedingungen
- Operator (analog STRIPS):
  - Liste von Anwendungsbedingungen (Preconditions)
  - Liste von beendeten Bedingungen (Delete-List)
  - Liste von beginnenden Bedingungen (Add-List)
- Suche mit Rückwärts-Verkettung bezogen auf Problemzerlegung (Liste von Teilzielen)
- Alternative Operatoren zur Erreichung von Teilzielen

Verzweigungsmöglichkeiten (Entscheidungspunkte) jeweils für

- Auswahl eines Teilziels
- Auswahl eines Operators
- Variablenbindungen eines Operators
- Anwendung eines Operators
- Zerlegung in Teilziele

Erweiterung von NOLIMIT durch PRODIGY/ANALOGY:

Suchverfahren an den Entscheidungspunkten fallbasiert steuern

Beispiel:

## Objekte

airplane: pl1

truck: tr9

objekt: obj4, obj7

place: p3 (post office), a3 (airport)

city: c3

## Bedingungen:

(at-obj object place)

(at-truck truck place)

(at-airplane airplane place)

(inside-truck object truck)

(inside-airplane object airplane)

## Ausgangszustand

(at-obj ob4 p3)

(at-obj ob7 a3)

(at-truck tr9 a3)

(at-airplane pl1, a3)

(same-city a3 p3)

## Ziel

(inside-truck obj4 tr9)

Operatoren	Pre-Cond.	Delete-List	Add-List
(load-tr object truck place)	(at-obj object place) (at-truck truck place)	(at-obj object place)	(inside-truck object truck)
(load-airplane object truck place)	(at-obj object place) (at-truck airplane place)	(at-obj object place)	(inside-airplane object airplane)
(drive-tr truck place1 place2)	(at-truck truck place1) (same-city place1 place2)	(at-truck truckt place1)	(at-truck truck place2)
(unload-airplane object airplane place)	(inside-airplane object airplane)	(inside-airplane object airplane)	(at-obj object place)
(unload-truck object truck place)	(inside-truck object truck)	(inside-truck object truck)	(at-obj object place)

## Indizierung/Retrieval/Auswertung

Basis für Ähnlichkeit von Fällen:

Startzustand (Liste von Bedingungen)

Ziel (Liste von Bedingungen)

Aufspaltung:

Ziel in “Unabhängige Teilziele”

(jeweils Konjunktion zusammengehörender Bedingungen)

Startzustand in “schwächste Initialbedingungen”

bzgl. (Teil-)Zielen

**footprint:** Minimale Initialbedingungsmenge eines (Teil-)Ziels  
– erzeugen durch rekursive Rückverfolgung (erklärungsbasiert)

*Teilziele* separieren durch Analyse des Plans

(partielle Ordnung berechnen:

zusammenhängende Komponenten bestimmen Teilziele)

Fälle mit gleichem Ziel  
haben oft unterschiedliche Initialbedingungen

### **Indizierung** der Fälle gemäß

- Zielen
- Initialbedingungen

1. Retrievalschritt:  
Übereinstimmung bzgl. Zielen mittels Hash-Technik
2. Retrievalschritt:  
Übereinstimmung bzgl. Initialbedingungen  
mittels Discrimination Network (Entscheidungsbaum)
3. Relevanz-Ordnung mittels Ähnlichkeitsmaß:

$$s = s_Z + s_I$$

wobei  $s_Z$  die Ähnlichkeit bzgl. der *Teilziele*,  
und  $s_I$  die Ähnlichkeit bzgl. der *footprints* mißt.

Schrittweise **Auswertung** unter Kombination von

- Lösungsschritten aus relevanten Fällen
- anderweitige Lösungsstrategien (z.B. MEA)

# 10. Technologische Fragen

Entwurfsentscheidungen bei Entwurf von FBS-Systemen:

- Festlegung der Indizes, z.B.
  - Merkmale (Attribute) mit Wertebereichen
  - Schlüsselwörter
  - IEs
  - Strukturen
- Festlegung des Fallformats
- Festlegung von Ähnlichkeit
- Festlegung des Retrievalverfahrens
- Festlegung des Adaptionverfahrens

Freiheiten beim Entwurf:

Container-Modell nach M.M. Richter

## 4 Wissens-Container

- Vokabular (Wissensrepräsentation für Fälle)
- Fallbasis
- Ähnlichkeit
- Adaption

Extremfälle z.B.:

- *Fallbasis* enthält *alle* Fälle,  
Beschreibung mit vollständigem *Vokabular*  
⇒ Identität als *Ähnlichkeit*, keine *Adaption*.
- *Fallbasis* enthält (k)einen Fall  
⇒ *Adaption* = Lösung “from scratch”.

Verschiebungen z.B.:

viele Fälle/geringer Ähnlichkeitsumfang  
vs. wenig Fälle/großer Ähnlichkeitsumfang

viele Fälle/wenig Adaption  
vs. wenig Fälle/viel Adaption

gute Ähnlichkeit/wenig Adaption  
vs. schlechte Ähnlichkeit/viel Adaption

## Grundproblem bezüglich Ähnlichkeit/Distanz im Fallbasierten Schließen:

1. Ähnlichkeit/Distanz impliziert Nützlichkeit.
2. Ähnlichkeit/Distanz basiert auf a-priori-Fakten, geringer Berechnungsaufwand.
3. Ähnlichkeit/Distanz als quantitatives Maß.

Dafür

- Auswahl geeigneter Merkmale.
- Auswahl geeigneter (intuitiver) Maße für Ähnlichkeit/Distanz.  
Zum Beispiel kompositorisch:
  1. Lokale Ähnlichkeiten/Distanzen,
  2. Kompositionsfunktion,
  3. Akkumulationsprinzip  
(Akzeptanz vs. Zurückweisung),
  4. Globale/lokale Monotonie unterstützen Intuitivität

# Pflege des Systems

## Aktualisierungen

- Kurzfristig:  
Neue Fälle aufnehmen (idizieren, Retrieval)
- Langfristig:  
Indizes, Fallformat, Ähnlichkeit, Retrieval, Adaption
- Vergessen:  
Irrelevante/redundante Fälle streichen.

## **Integration weiterer Techniken**

- gegenseitige Ergänzung:
  - allgemeines = Regelwissen
  - spezifisches (Ausnahmen usw.) = Fallwissen
- Hintergrundwissen
  - für Ähnlichkeit, Adaption
  - Modellwissen, Simulation, ...
- Lerntechniken:
  - für Ähnlichkeit
  - Statistik, Neuronale Netze

## **Beziehung zum Maschinellen Lernen:**

Fallbasierte Systeme:

- „Idealfall lernender Systeme“:
  - Wissenserwerb mit der Arbeit
  - Erfahrungen direkt auswerten
- Grenzen:
  - Abstraktion
  - Generalisierung
- dem Spezifischen verhaftet

Kombination FBS und ML: INRECA

## **Vorteile FBS:**

- unmittelbare Darstellung von Zusammenhängen
  - schnelle Auswertung
  - Protokoll bekannter Lösungen, bekannter Fehler
  - weitgehend selbsterklärend
- einfache Akquisition
  - existierende Datensammlungen ausnutzen
- flexibel
- lernfähig
- kognitiv adäquat

## **Erste Tools Anfang der 90er**

REMIND (vorher: CBR Shell) von Cognitive Systems

CBR Express von Inference Corp.

ESTEEM von ESTEEM

## **Neue Entwicklungen:**

CBR-Works von tecInno (Kaiserslautern)

ORENGE von tecInno (Kaiserslautern)

u.a.