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,
- ungefährer 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

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

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

und vorgegebener "Bucketgröße" b:

- Falls $d = card(C) \leq b$: Blattknoten, der C repräsentiert.
- Falls $card(C) > b$:
  - Wähle ein **günstiges** *Diskriminatorattribut* $i \in \{1, ..., k\}$ 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 $C_\geq$.

Ein Unterbaum repräsentiert (zunächst) die zur Konstruktion verwendete Menge $C_<$ bzw. $C_\geq$.

"Primärer" Retrievalprozeß wie üblich:
Für Anfrage $q = [q(1), ..., q(k)]$ wird jeweils beim Test "$q(i) < p$?"

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

(bei unbekannten Werten: beide Unterbäume weiter verfolgen)

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

Primäres Retrieval für eine Anfrage $q = [q(1), ..., q(k)]$
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 gemäß Nähe zu q sortieren: $g_1, ..., g_n, ...g_b$ (Annahme: Bucketgröße $b \geq n$, sonst Nachbarintervalle einbeziehen) $g_1, ..., g_n$ 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 $g_1, ..., g_n$.
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 (vgl. Lernen von Entscheidungsbäumen, z.B. C4.5):
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 → Zurückweisug.
Nächster Nachbar: *alle* lokalen Werte sind nah.

Methode: Entscheidungsbaum
("top down pruning")

- Distanz:
  akkumuliert *Argumente für Ausschluss*

Akkumulation für Zurückweisung

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: $r_2, r_3, ...$ bis keine Verbesserung mehr möglich.

mögliche Probleme:
Zyklen, lokale Maxima,
weitere ähnliche Fälle in C.

5. Anpassung (Adaptation)

Übertragung auf ähnliche Fälle:

- Anpassung an aktuelles Problem
  - Interpolation (stetiger Problemlösungsraum)

  Anpassungsformeln bzw. -regeln:
  Abweichung der Probleme $\Rightarrow$ Abweichung der Lösung

- Kombination der Lösungen mehrerer Fälle

Beipiel 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,-
...						

Beipiel 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.-18.8.07	2	****	3000,-
2	Potsdam	Kultur	20.-25.8.07	2	Zelt	500,-
3	Edinburgh					
...						

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	...		

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

Verschiebung zwischen Container Fälle und Container Anpassung:
Mehr Fälle – weniger Anpassung

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 oder veralteter Fälle

Verfahren abhängig von Domäne:

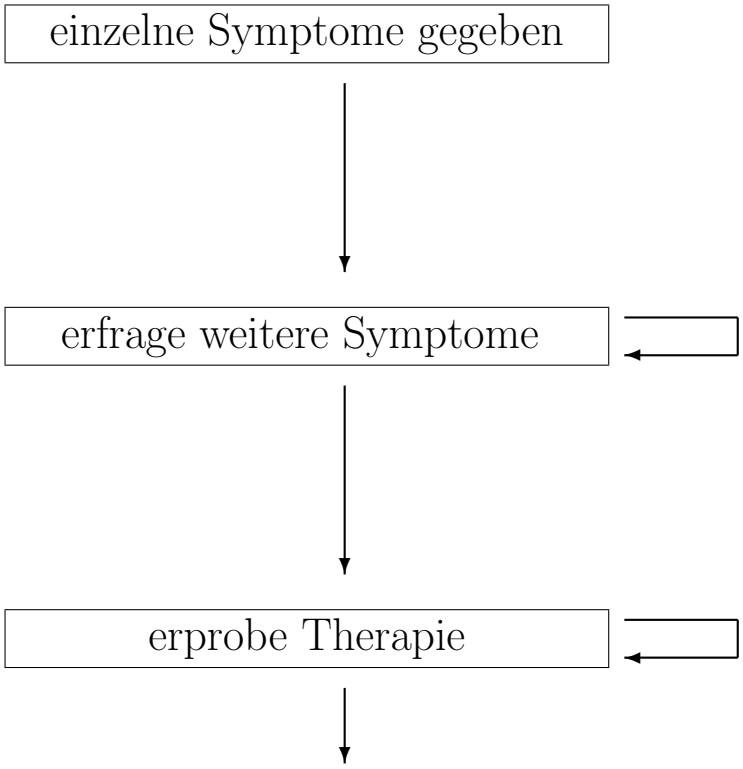
- Katalog (Last-Minute-Reisen): automatisch
- Diagnose-Dokumente
  - Einfügen neuer Begriffe durch Nutzer (mit System-Unterstützung)
  - Einfügen neuer Dokumente: automatisch

7. Erweiterte Betrachtung: Fallvervollständigung

Klassisch: Fall = Problem + Lösung: P + L

⇒ Regelartig: Wenn Problem, dann Lösung.

Dagegen: Diagnose als *Prozess*



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 Informationssammlung

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. Abgeschlosssene 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: Diese Bedingungen gehören zusammen.
Verallgemeinerung: Diese Bedingungen gelten immer zusammen.

Abfrage analog zu Datenbanken oder Prolog:

Einzelne Fakten eines Falles gegeben
- welche Fakten können dazu gehören.

Probleme als „unvollständige“ Beschreibungen von Fällen.
Gemeinsame Form: Anfragen und Fälle sind

$$\boxed{\textit{Mengen von Informationseinheiten (IE)}}$$

$$q = q_1, ..., q_k$$

$$c = c_1, ..., c_n$$

- 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

$\Rightarrow$ 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),*

σ*:* Ähnlichkeitsfunktion

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

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

ρ*:* Relevanz-Funktion

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

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

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

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

Weitere Angaben z.B.: Die Funktionen π_n sind monoton.

„Zustand“ eines Netzes:

Definition

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

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

„Standard Form“ einer Anfrage:

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

Speziellere Anfragen:
α_{query} mit speziellen („Wichtigkeits“-)Werten für spezielle IE.

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

Propagierung:

Definition

Gegeben: BCRN $N = [E, C, \sigma, \rho, \Pi]$ *with* $E = \{e_1, ..., e_s\}$.
$\alpha_t : E \to \mathcal{R}$ *sei Aktivierung zur Zeit* t.
Die Aktivierung von IE-Knoten $e \in E$ *zur Zeit* $t+1$ *ist gegeben durch*

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

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

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

Restriktion auf 2 Schritte:

Schritt 1 :

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

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

Schritt 2 :

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

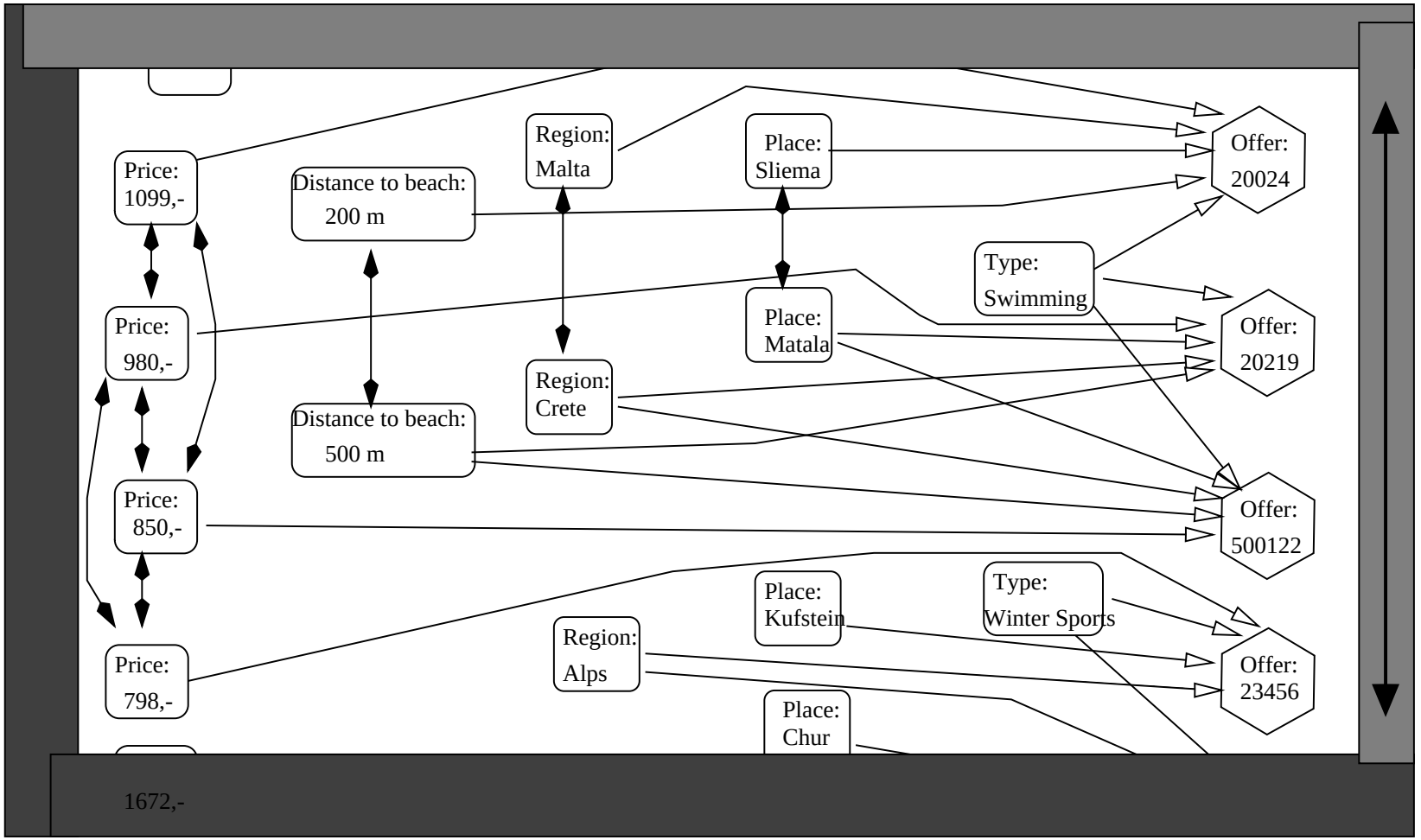
$$\alpha_2(c) = \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), ..., \rho(e_s, c) \cdot \alpha_1(e_s)).$$

Resultat

Gegeben: BCRN $N = [E, C, \sigma, \rho, \Pi]$ *mit* $E = \{e_1, ..., e_s\}$.
Die resultierende Aktivierung $\alpha_N : C \to \mathcal{R}$ *von Fall-Knoten bezüglich einer Anfrage* $\alpha_{query} : E \to \mathcal{R}$ *ist gegeben (nach 2 Schritten) durch*

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

Das Resultat des Fall-Retrievals *bzgl. der Anfrage* α_{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)

IE-NODES **CASE-NODES**

RELEVANCE-ARCS (annotations $\rho(e,c)$ **omitted)**

SIMILARITY-ARCS (annotations $\sigma(e,e')$ **omitted)**

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 $e_{(A,w)}$

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

   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:

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

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

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

gegeben durch

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

für den kein IE-Knoten $e_{(A,w)}$ in E_A existiert.

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

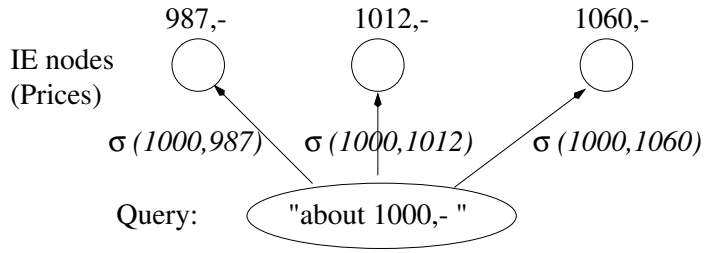
Berechnung der Ähnlichkeiten:

- 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.

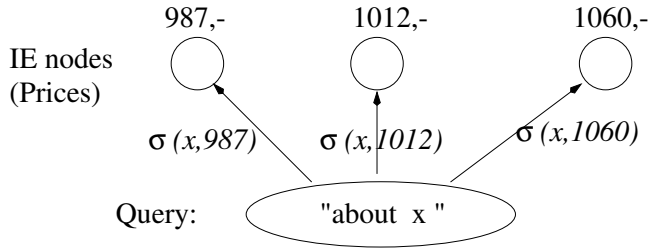
  alternativ:

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

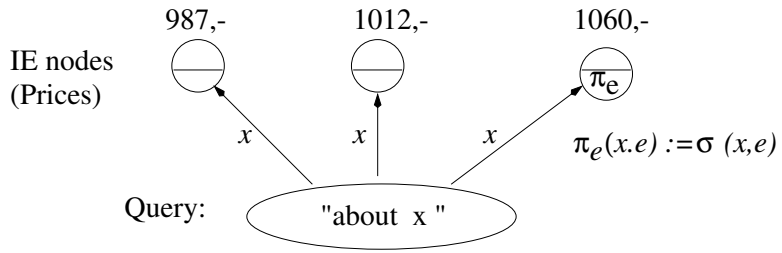
Beispiel für Implementation von fehlenden Werten:



Intended similarity



Intended similarity (general case)



Implementation with propagation functions in IE nodes

Kompositorische Ähnlichkeitsfunktionen in BCRN

Satz

Jede kompositorische Ähnlichkeitsfunktion σ kann durch ein BCRN $N = [E, C, \sigma, \rho, \Pi]$ berechnet werden:
Sei $c_{[c_1,...,c_n]} \in C$ der Falldeskriptor für den Fall $[c_1, ..., c_n]$,
und sei $[q_1, ..., q_n]$ eine Anfrage.
Wenn $[q_1, ..., q_n]$ in die Anfrageaktivierung

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

transformiert wird, so berechnet das BCRN:

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

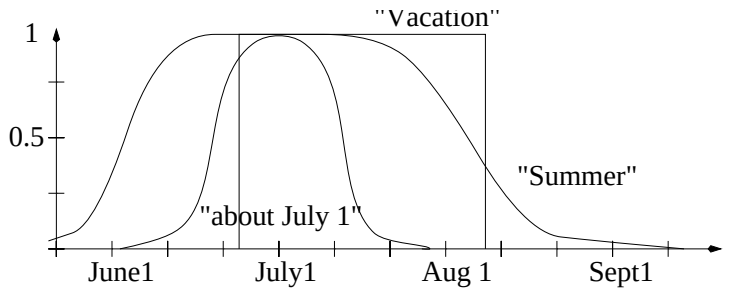
Lokale Ähnlichkeiten (Vage/fuzzy Begriffe):

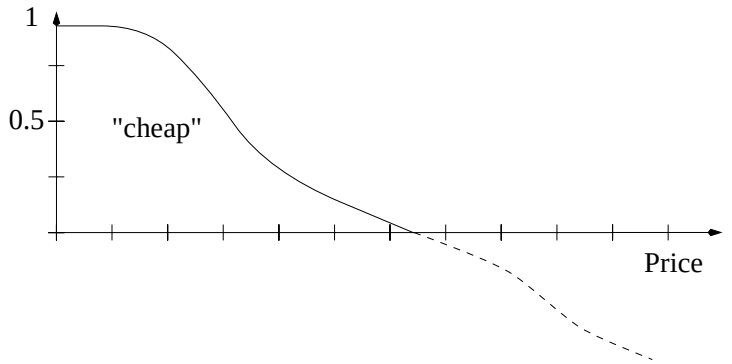
- Linguistische Terme
- Vage Beschreibung konkreter Werte („ungefähr 2000 DM“)

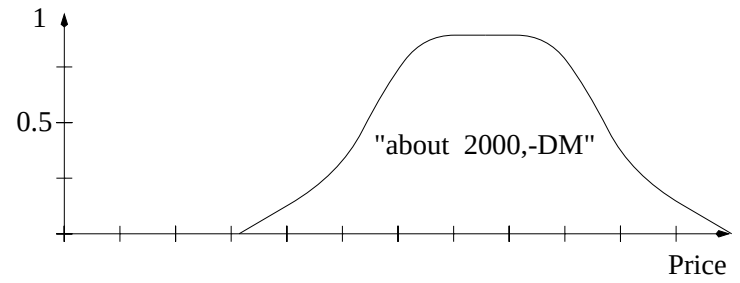
Interpretation von Fuzzy-Mengen
(Charakteristischer Fkt.)
für das Retrieval („**erinnern**“):

“Wichtigkeit beim Wert x an
den lingustischen Term zu denken”

(Negative Werte: Nicht erinnern)

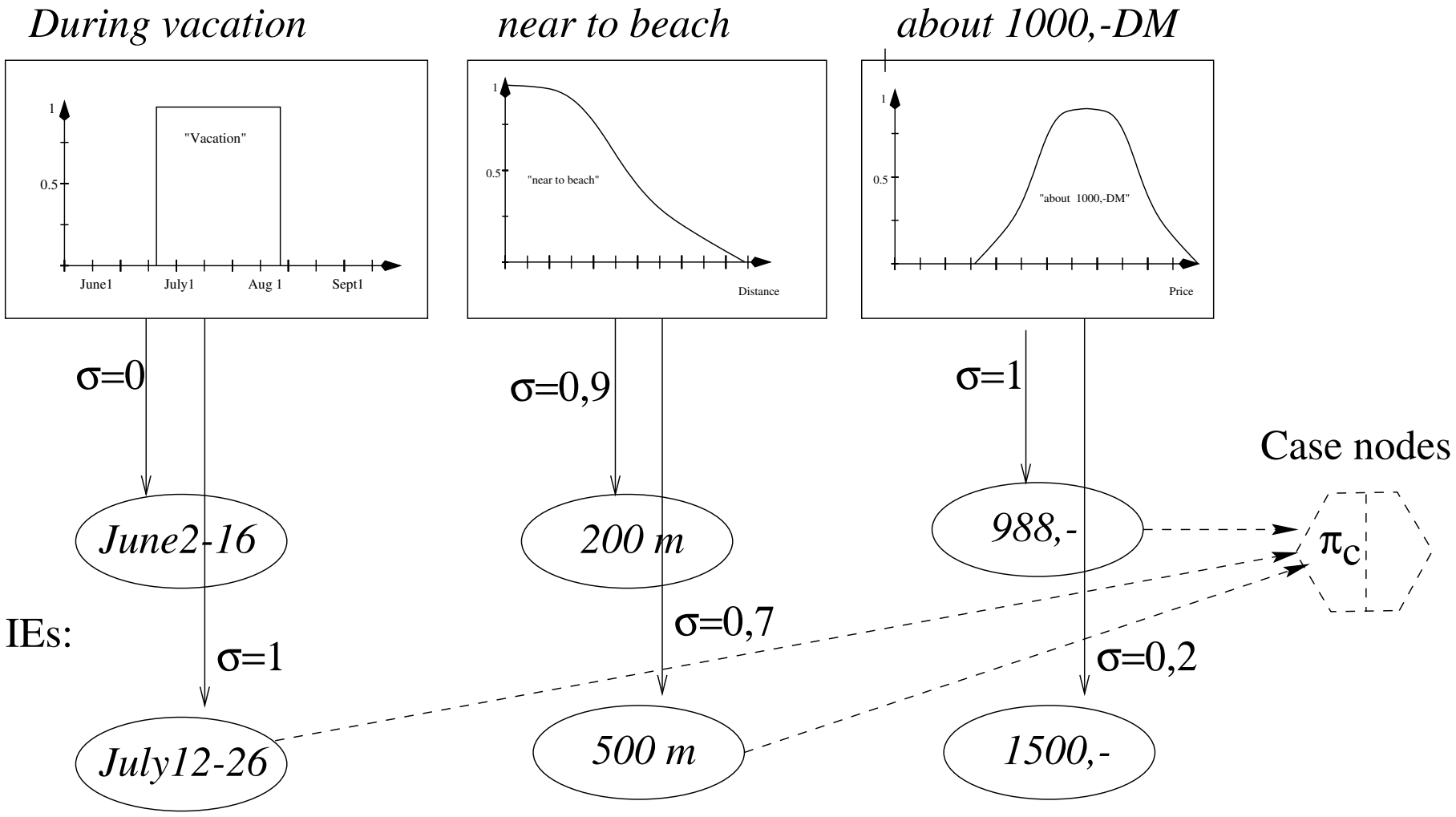






Komposition:

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 π_c (z.B. Σ, Max, Min, ...)

$\rightarrow$ 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 π_e (z.B. im Text-Retrieval)

Spezielle Möglichkeiten/Eigenschaften:

- modulare Struktur
- leichte Veränderbarkeit
- unterschiedliche Arten von IE
- Ähnlichkeit zwischen beliebigen IE
- kompositorische Ähnlichkeitsfunktionen
- individuelle Ähnlichkeitsfunktionen für jedes $c \in C$.
- Variationen in der Anfrage-Aktivierung α_{query}
- „bottom-up“ Fall-Rekonstruktion (Fallvervollständigung) statt „top-down“ Ausschlussverfahren
- Effektive Retrieval-Prozedur
- Fehlende Werte unproblematisch
- Parallele Arbeit
- Brücke zu subsymbolischen Verfahren

Verwandte Modelle

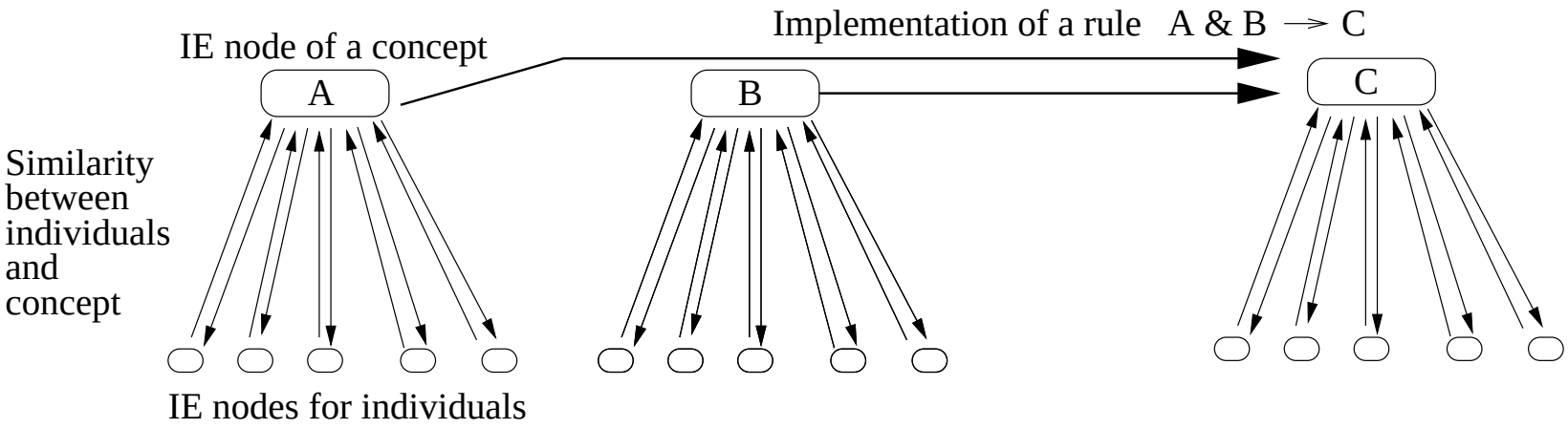
- Assoziative Speicher,
- Semantische Netze,
- Conceptual Dependency/Scripts/Dynamic Memory,
- Discrimination Networks,
- Neuronale Netze
  - rekurrent
  - strukturiert

Offenes Problem:
Lernverfahren von Neuronalen Netzen für Ähnlichkeiten

Erweiterungen

- Aktivierungs-Ausbreitung über mehrere Takte $\Rightarrow$ 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
- 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 als **horizontale** Konstrukte auf höherer Ebene.

Spreading activation Nets (SAN)

BCRN zunächst: Ausbreitung von Aktivierungen in 2 Schritten:
1. gemäß Ähnlichkeit (σ) zwischen IE-Knoten
2. gemäß Relevanz (ρ) von IE-Knoten zu Fall-Knoten

CRN allgemein:

	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	...
IE-Knoten:	$\alpha_0 = \alpha_{query}$	α_1	α_2	α_3	α_4	...
Fall-Knoten:		α_1	α_2	α_3	α_4	...

Berechnungen über mehrere Takte:

- „any-time-computation“ :
  Fortführung des Retrieval-Prozesses über mehrere Takte bis zu „befriedigender“ Lösung
- Einbindung von Konzepten usw.

Probleme: Zyklen, Stabilität, „Abklingen“

→ Simulation von SAN durch BCRN?

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

→ Einfügen „direkter Verbindungen“ (Hebbsche Regel)

CRN als kognitives Modell
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 *symbolischen* und *subsymbolischen* Zugängenn

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)

Lernen in CRN

mit verschiedenen *Zielen*

- neues Wissen (von außen erworben, intern erarbeitet)
- Anpassung an Probleme: „Ähnlichkeiten lernen“

mit verschiedener *Realisierung*

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

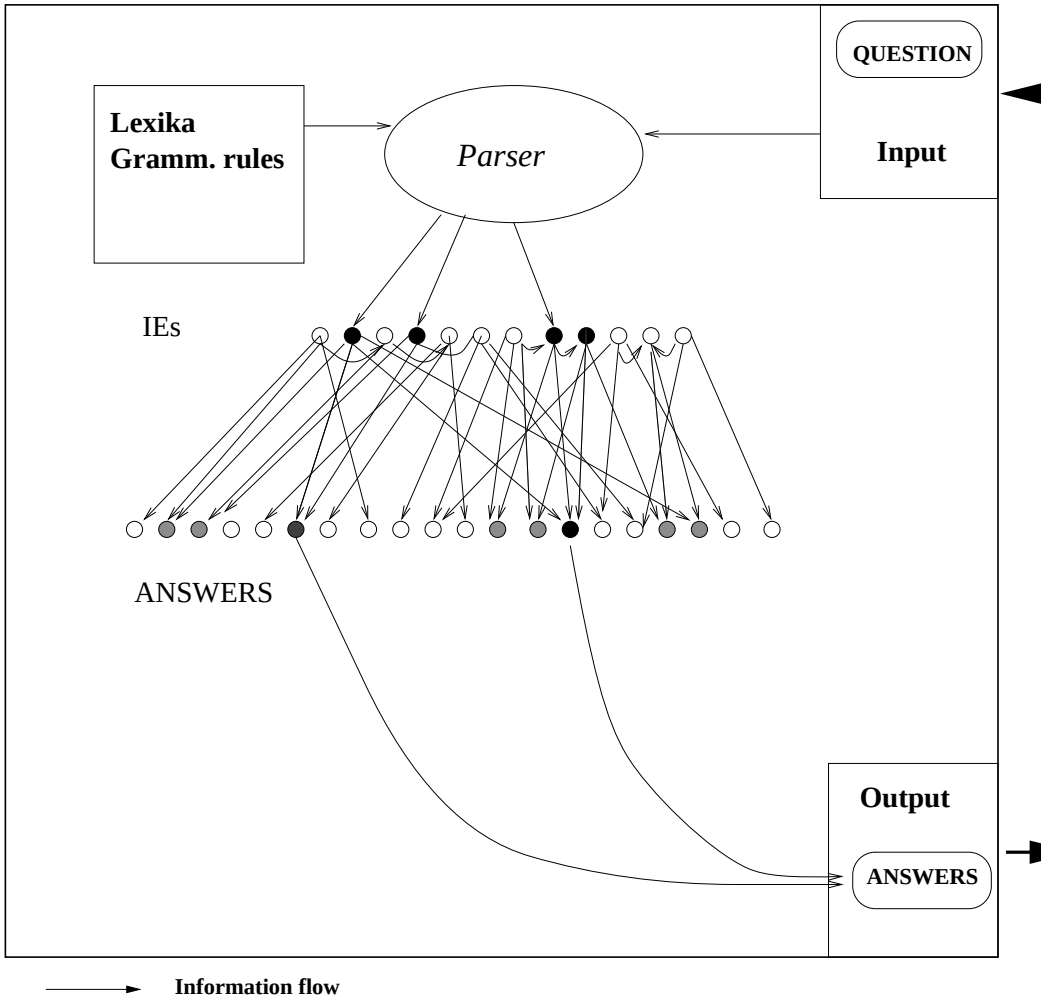
CRN für „flaches“ NLP: Retrieval von textuellen Dokumenten

IEs sind Konzepte

Wörter/Wortgruppen sind „Erscheinungsformen“ der Konzepte

(Grammatische Formen, Synonyme, andere Sprache, ...)

Ähnlichkeit zwischen Konzepten: Ähnlichkeit von IEs



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

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)

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

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

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 Adaptionsverfahrens

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*
  $\Rightarrow$ Identität als *Ähnlichkeit*, keine *Adaption*.
- *Fallbasis* enthält (k)einen Fall
  $\Rightarrow$ *Adaption* = Lösung "from scratch".

Verschiebungen z.B.:

viele Fälle/geringer Ähnlichkeitsumfang vs. wenig Fälle/groß er Ähnlichkeitsumfang

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

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