Einstieg

„Building Domain-Specific Search Engines with Machine Learning Techniques“

McCallum et al. 1999

- Vorgestellte Techniken
  - Reinforcement Learning
  - Naive Bayes Classification
  - Hidden Markov Models
    - Viterbi

Beispiele für Suchmaschinen

- **www.campsearch.com**

  Suche nach Sommercamps

- **www.netpart.com**

  Firmen suche

- **www.mrqe.com**

  Video Kritiken

- **www.math.usyd.edu.au:8000/MathSearch.html**

  Mathematische Inhalte

- **www.travelfinder.com**

  Urlaubsangebote

Aufbau einer Suchmaschine

3 Verschiedene Aufgaben

- Collecting
- Extracting
- Presenting

Cora

- gibt es leider nicht mehr

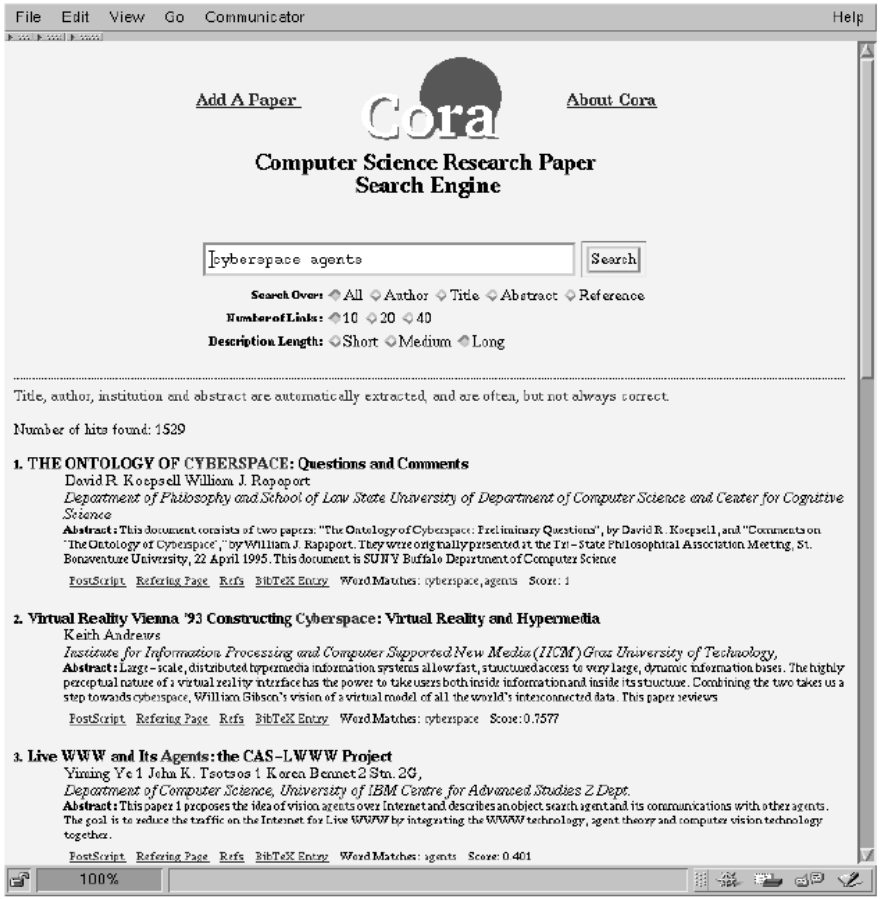


Figure 1: A screen shot of the query results page of the Cora search engine (*www.cora.jprc.com*). Note that paper titles, authors and abstracts are provided at this level.

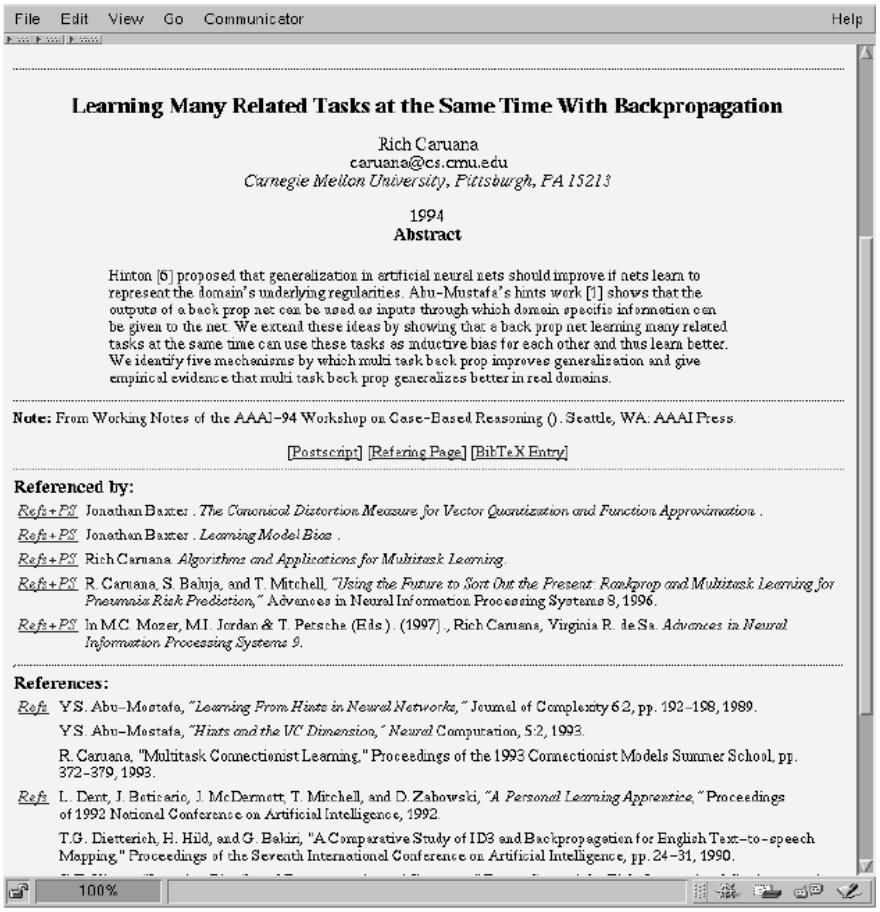


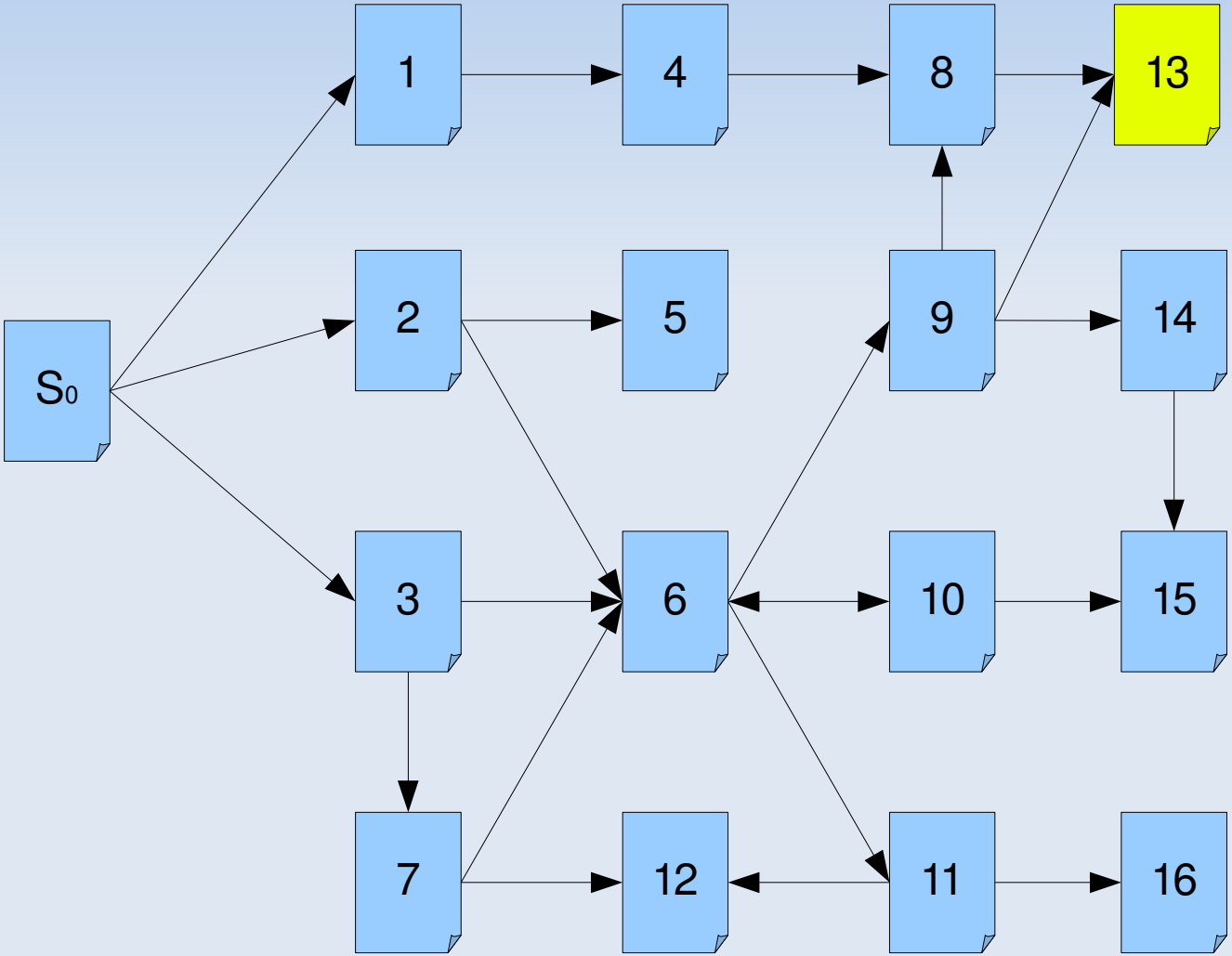
Figure 2: A screen shot of a details page of the Cora search engine. At this level, all extracted information about a paper is displayed, including the citation linking, which are hyperlinks to other details pages.

Methoden (Cora)

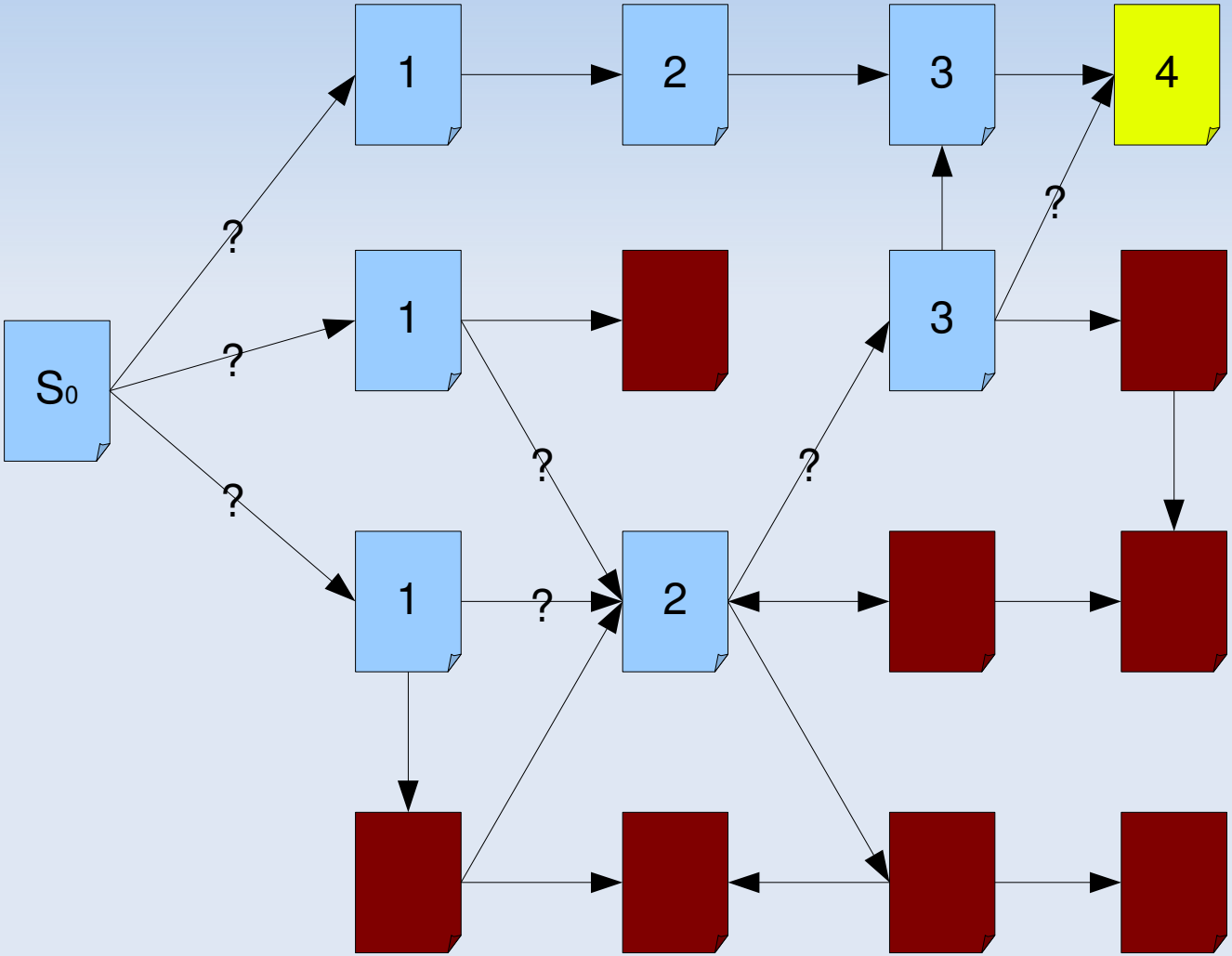
- Collecting durch spidering
  - Mögliche Ziele
    - Alle (bzw. möglichst viele) Dokumente finden (Google, Yahoo, ...)
      - Mittels Breitensuche
    - Bestimmte Dokumente finden (Cora)
      - Möglichst schnell (kurzer Pfad zum richtigen Dokument)
      - Möglichst genau

      Methode ?

Reihenfolge Breitensuche



„Optimale“ Reihenfolge(n)



Spidering

- Ziel: Suche die am „günstigsten“ Links aus

- Lernen durch Belohnung / Bestrafung
  - Allerdings: Keine Antwort über korrektes Verhalten
  - Sondern: skalare Belohnung !

Reinforcement Learning (Formal)

S : Menge der Zustände

A : Menge der „Aktionen“

T : Transitionen $T : S \times A \rightarrow S$

R : Belohungsfunktion $R : S \times A \rightarrow \mathbb{R}$

Gesucht wird ein optimaler Pfad $\pi : s_0 \rightarrow \ldots \rightarrow s_x$

Der Wert eines Zustandes für einen Pfad kann angegeben werden als:

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r_t$$

(t ist dabei der t-te Schritt, γ ist „Dämpfung“ $0 < \gamma < 1$)

Ein optimaler Pfad kann angegeben werden als:

$$Q^{opt}(s,a) = R(s,a) + \gamma V^{opt}(T(s,a))$$

Erfüllt Bellmansche Optimalitätsgleichung → Dynamische Prog.

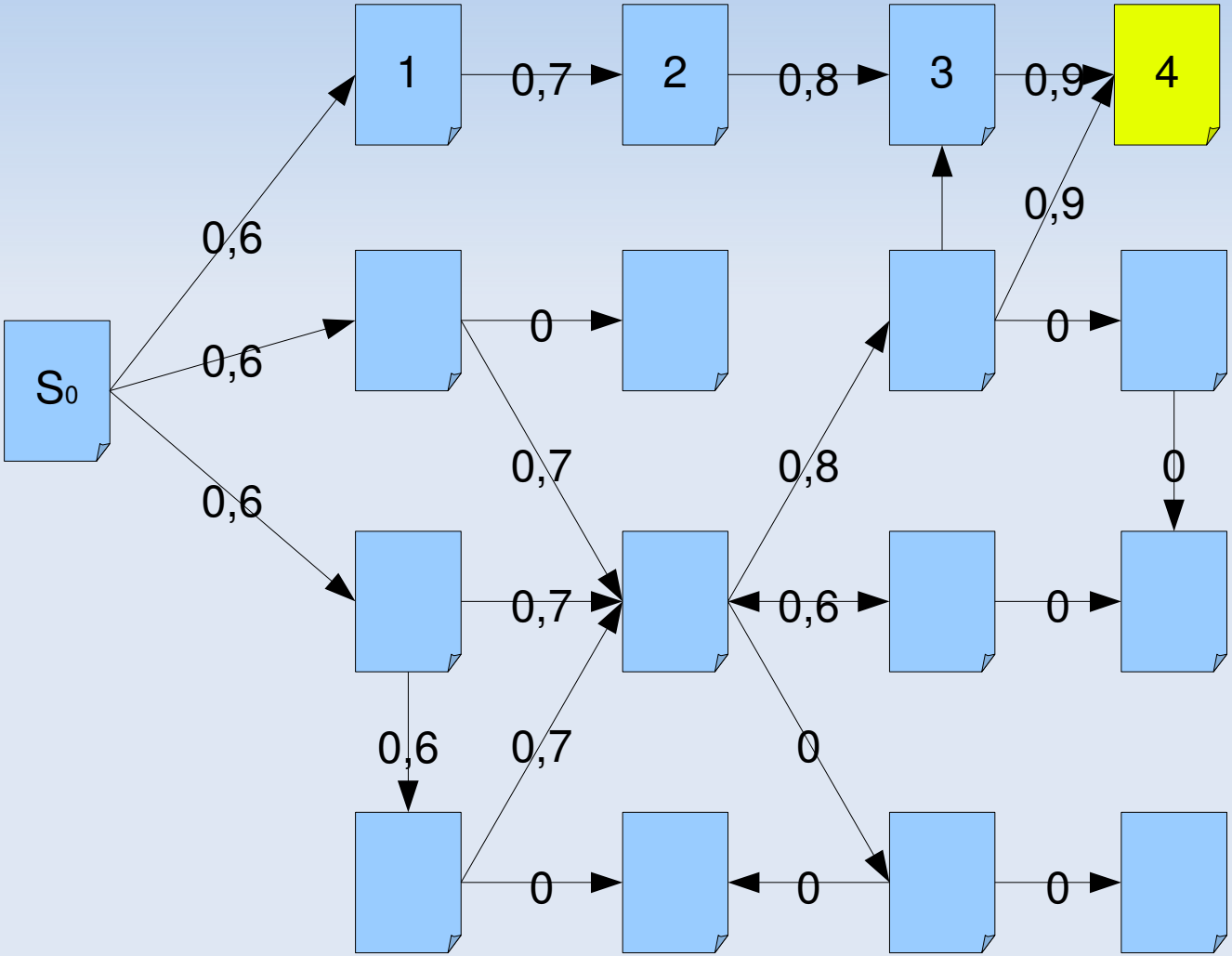
RL & Spidering

- S = Dokumente (gesuchte Dokumente)
- A = Links (repräsentiert durch bow *1)
- R = Belohnung durch richtige Klassifikation

- (*1 bag of words / Vektor aus Anzahl der Worte)

der,0
die,0
das,1
lernen.1
...

Idee Reinforcement Learning



RL & Spidering

Und was wird jetzt „gelernt“ ?

- welche Art Links erfolgversprechend sind

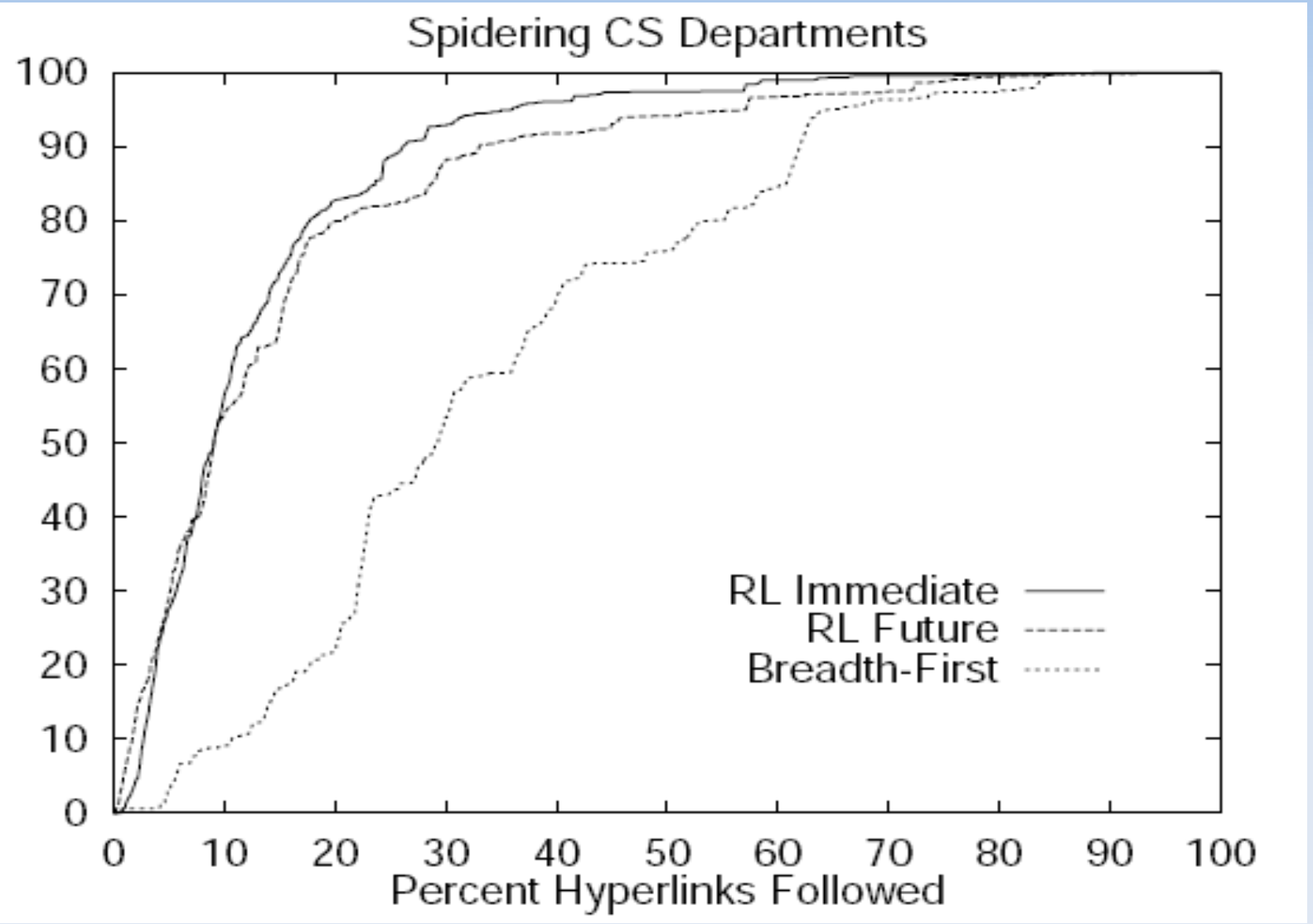
Wie wird „gelernt“ ?

- Durch Bewertung von Wort-Vektoren

RL & Spidering - Test

- Basisdaten (Webspace 4 Universitäten)
  - Brown University / Cornell University / University of Pitsburgh / University of Texas
- Jeweils 3 Zum Trainieren 1 zum Ausprobieren
  - (Also 4 Durchgänge)
- jeweils noch mal mit $\gamma = 0$

RL & Spidering (ergebnisse)



(„Building Domain Specific Search Engines with Machine Leaning Techniques“, 1999 McCallum et al,, S. 5 (3.4 - Figure 3))

RL & Spidering (ergebnisse)

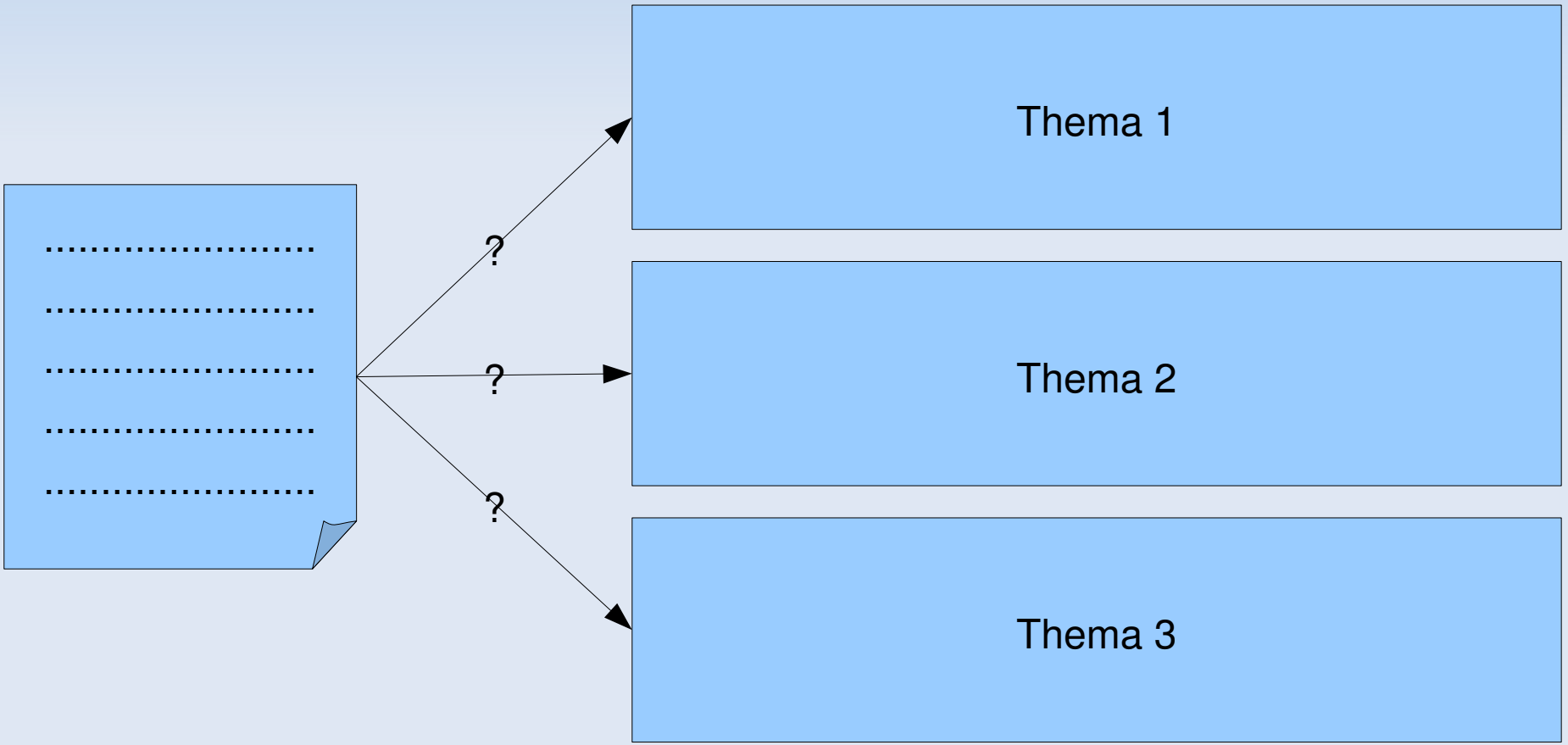
„One Measure of Performance is the number of hyperlinks followed by before 75% of the research papers are found. Reinforcement learning performance is significantly more efficient, requiring exploration of only 11% of the hyperlinks, in comparison to the breath-first searchs 30%. This is nearly a factor of three increase in spidering efficiency“

(„Building Domain Specific Search Engines with Machine Leaning Techniques“,

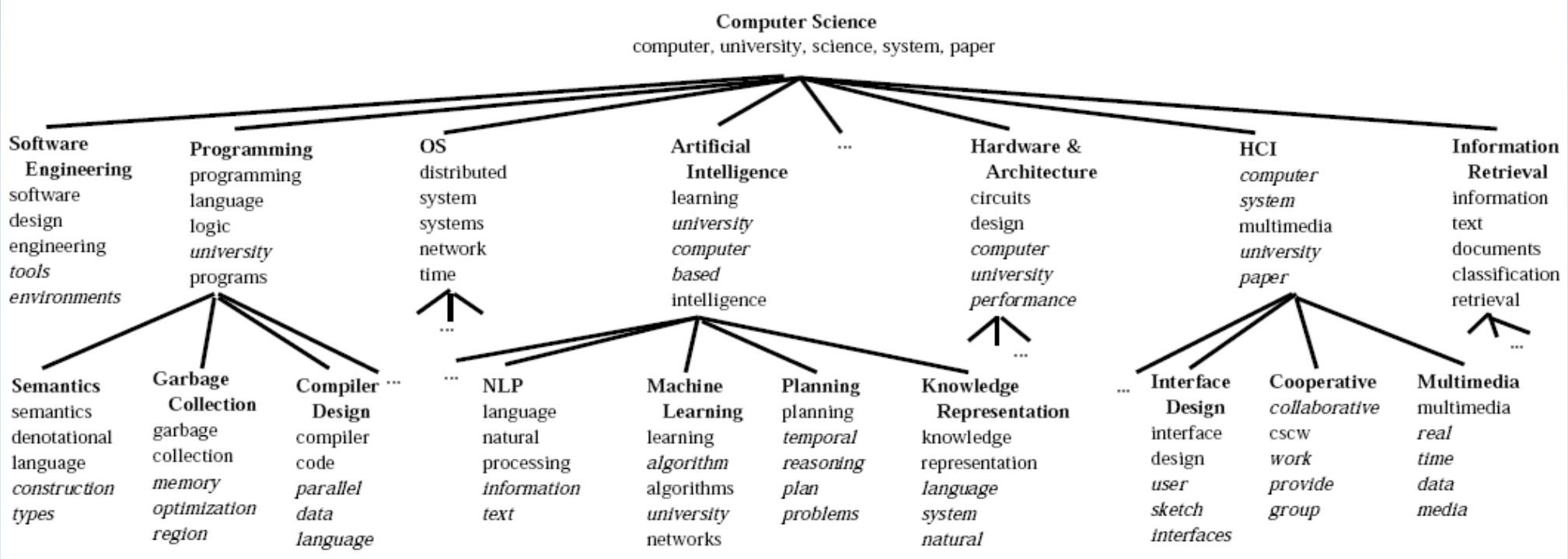
1999 McCallum et al,, S. 5 (3.4))

Klassifikation

Wie kann man ein Dokument einem Thema zuordnen ?



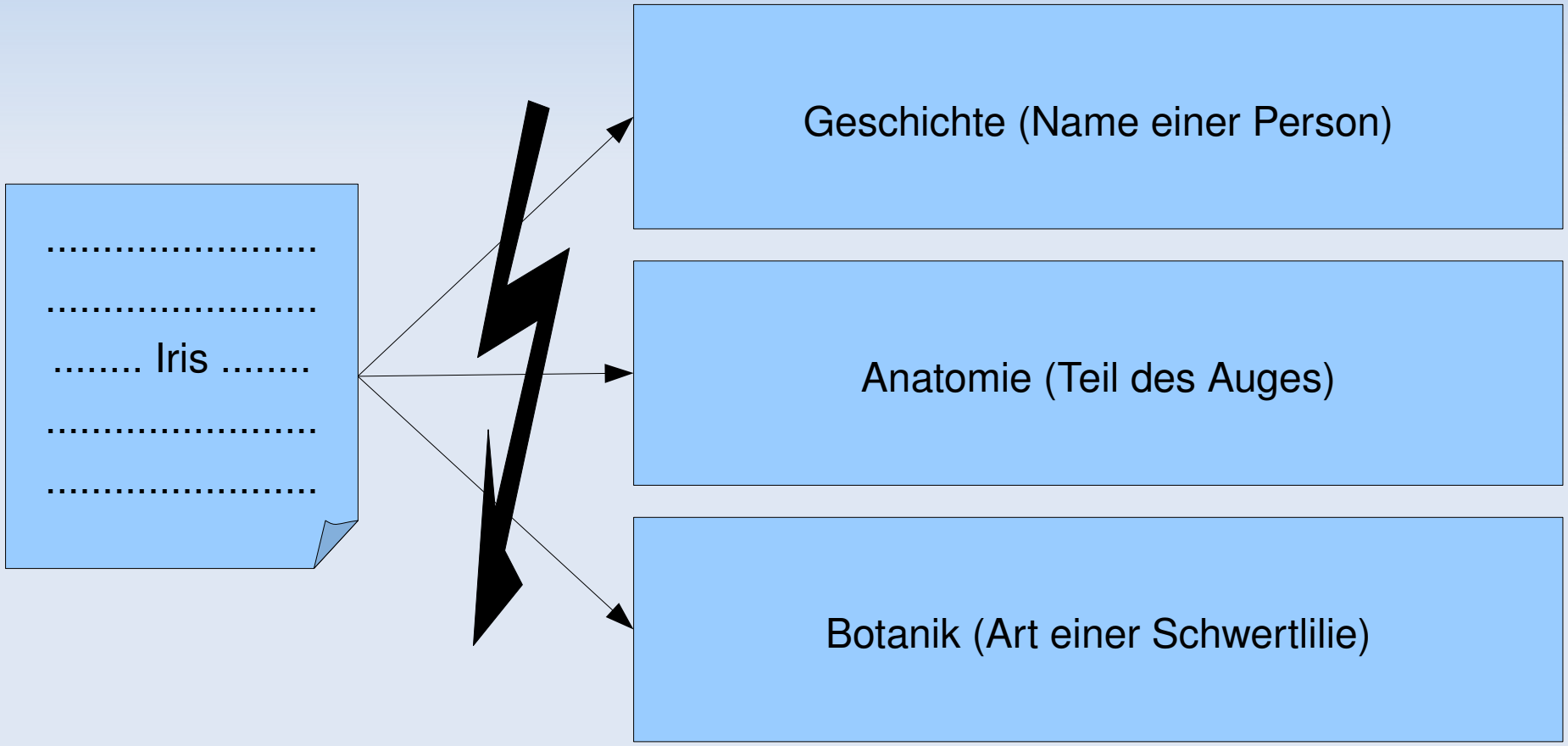
Klassifikation Cora



(„Building Domain Specific Search Engines with Machine Leaning Techniques“, 1999 McCallum et al,, S. 6 (3.5 - Figure 4))

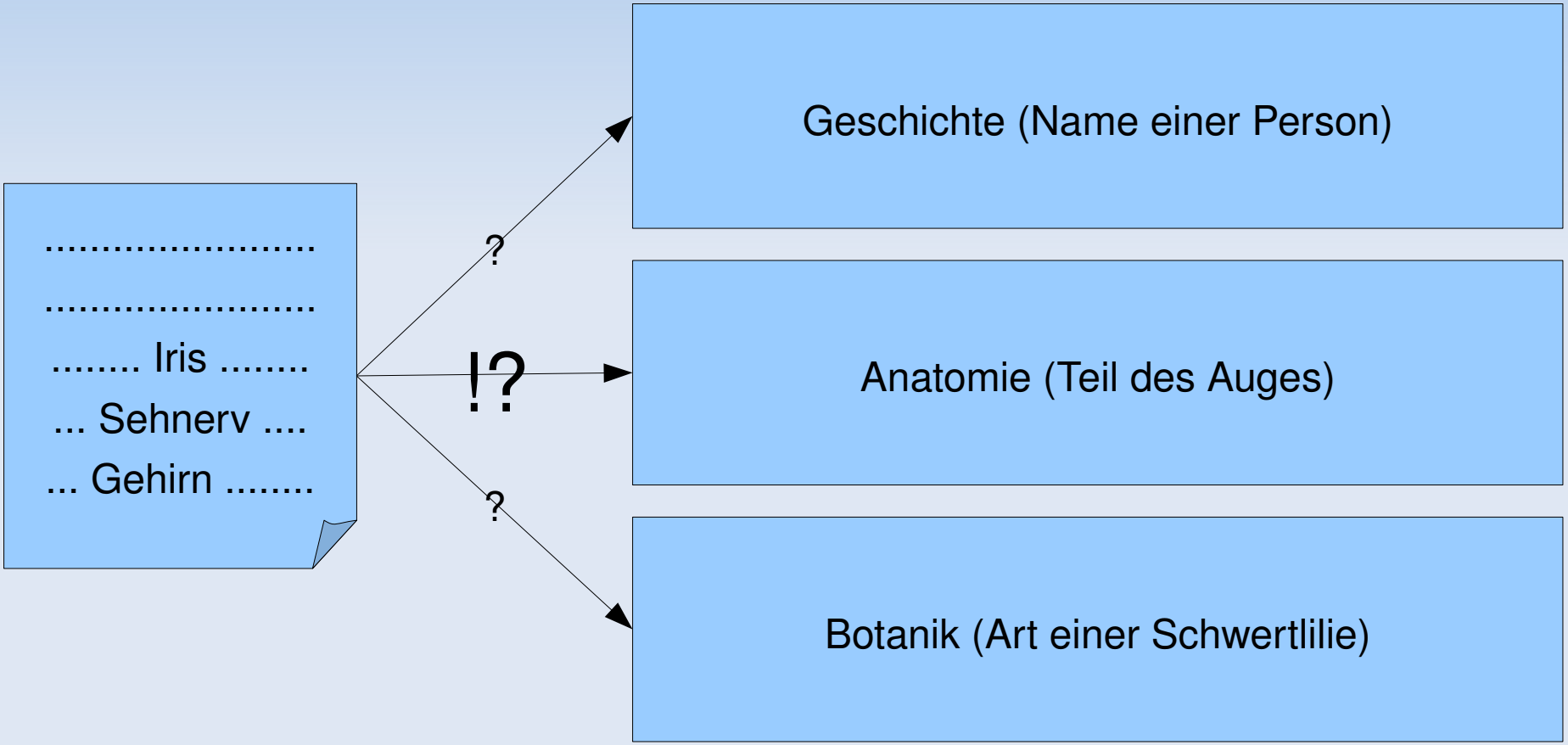
Klassifikation

Schlüsselwörter



Klassifikation

Kombination von Schlüsselwörtern



Mögliches Problem: Welche Schlüsselwörter nehme ich ?

(Vollständigkeit)

Klassifikation

- mögliche Probleme
  - Auswahl der Schlüsselwörter
  - Fehlende / Neue Schlüsselwörter
- Ausweg
  - Klassifizierte Trainingsmenge
    - generiere daraus „bag of words“
  - Bayes

Biologie
Gehirn, 5
Iris, 10
Sehnerv. 5
gießen, 0
...

Botanik
Gehirn, 0
Iris, 5
Sehnerv, 0
gießen, 3
...

Stochastik (Bayes)

Bayes

$$P(A|B)=\frac{P(B|A)\,P(A)}{P(B)}$$

Beispiel (grobe Schätzungen, und stark vereinfacht)

S: Schwedin

B: Blond

$$P(S)=\frac{9\text{M}}{6625\text{M}}\approx 0.0014 \qquad P(B)=0.05 \qquad P(B|S)=0.7$$

$$P(S|B)=\frac{P(B|S)\,P(S)}{P(B)}=\frac{0.7\cdot 0.0014}{0.05}\approx 0.02$$

Stochastik (Bayes)

- Geht das überhaupt ?
  - Keine Ahnung wie die Welt wirklich ist !!!!
    - Wäre aber notwendig um Wahrscheinlichkeiten anzugeben ... (zumindest streng genommen)
- Macht man trotzdem
  - (K)einer weiß wie die Welt wirklich ist !!!!
  - Je besser die Trainingsdaten desto näher an der Welt
  - Praktikabel

Annahmen

- Jedes Thema „erzeugt“ bestimmte Wörter in einer bestimmten Frequenz. (kommen in einer bestimmten Frequenz in einem Text vor)
- Durch Bayes kann man dann bestimmen „welches Thema die Daten am wahr-scheinlichsten „erzeugt“ hat.
- **Die Naive Annahme** dabei ist, dass die Worte unabhängig von ihrer Position und den übrigen Worten sind.

Naive Bayes (Formal)

C : Menge der Dokumente (Klassen/Topics)

W : Menge der Wörter

D : Menge der Dokumente

Wahrscheinlichket des Vorkommens der Klasse j	$P(c_j)$
Wahrscheinlichkeit das ein Wort i in Klasse j vorkommt	$P(w_t \mid c_j)$
Wahrscheinlichkeit Der Klasse j bei Dokument i	$P(c_j \mid d_i)$

$$P(c_j \mid d_i) \propto P(c_j) P(d_i \mid c_j) \propto P(c_j) \prod_{k=1}^{|d_i|} P(w_{d_{ik}} \mid c_j)$$

Naive Bayes (Formal 2)

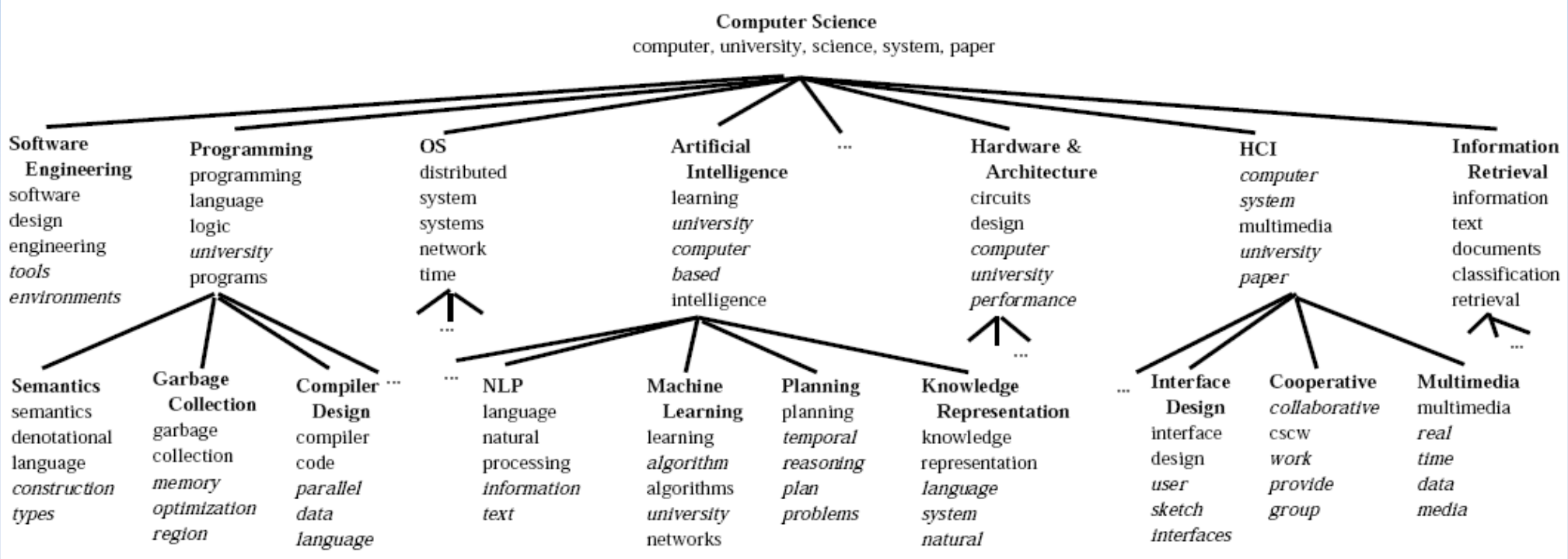
$$P(w_t|c_j)=\frac{1+\sum_{d_i\in D}|\{w_t|w_t \text{ in } d_i\}|P(c_j|d_i)}{|W|+\sum_{s=1}^{|W|}\sum_{d_i\in D}|\{w_s|w_s \text{ in } d_i\}|P(c_j|d_i)}$$

$$P(c_j)=\frac{1+\sum_{d_i\in D}P(c_j|d_i)}{|C|+|D|}$$

kleines Problem

- Instabil wenn:
  - Qualität der Trainingsdaten nicht „gut“ ist
    - Sie also das Vorkommen der Wörter in der realen Welt nicht gut genug widerspiegeln
- Verbesserung:
  - Shrinkage

Klassifikation Cora



(„Building Domain Specific Search Engines with Machine Leaning Techniques“, 1999 McCallum et al,, S. 6 (3.5 - Figure 4))

Naive Bayes (Shrinkage)

Ziel:

Verbesserung der Genauigkeit der Klassifikation.

Wie kommt man jetzt dahin ohne die Wörter und ihre Verteilung genau zu kennen?

IDEE:

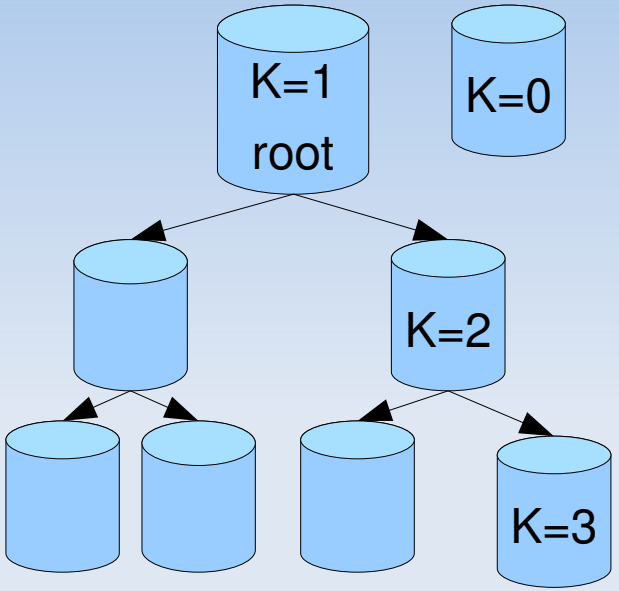
Ziehe alle Elternklassen mit in Betracht

Damit landet ein Dokument in dem wahrscheinlichsten Bereich.

(Dadurch das auch das überige Vokabular auch am wahrscheinlischsten aus dem übergerordneten Themenbereich stammt)

Naive Bayes (Shrinkage vorgehen)

$$\check{P}(w_t|c_j) = \lambda_j^1 P^1(w_t|c_j) + \ldots + \lambda_j^k P^k(w_t|c_j)$$



Wie die λ für den Pfad bestimmen ?
Bestimme für jedes Wort die WK das es vom i-ten Elterndokument erzeugt wurde

$$\beta_j^i = \sum_{w_t \in d_i \in D} \lambda_j^1 P^i(w_t|c_j) P(c_j|d_i)$$

Normalisiere mit

$$\lambda_j^i = \frac{\beta_j^i}{\sum_{i=1}^{k} \beta_j^i}$$

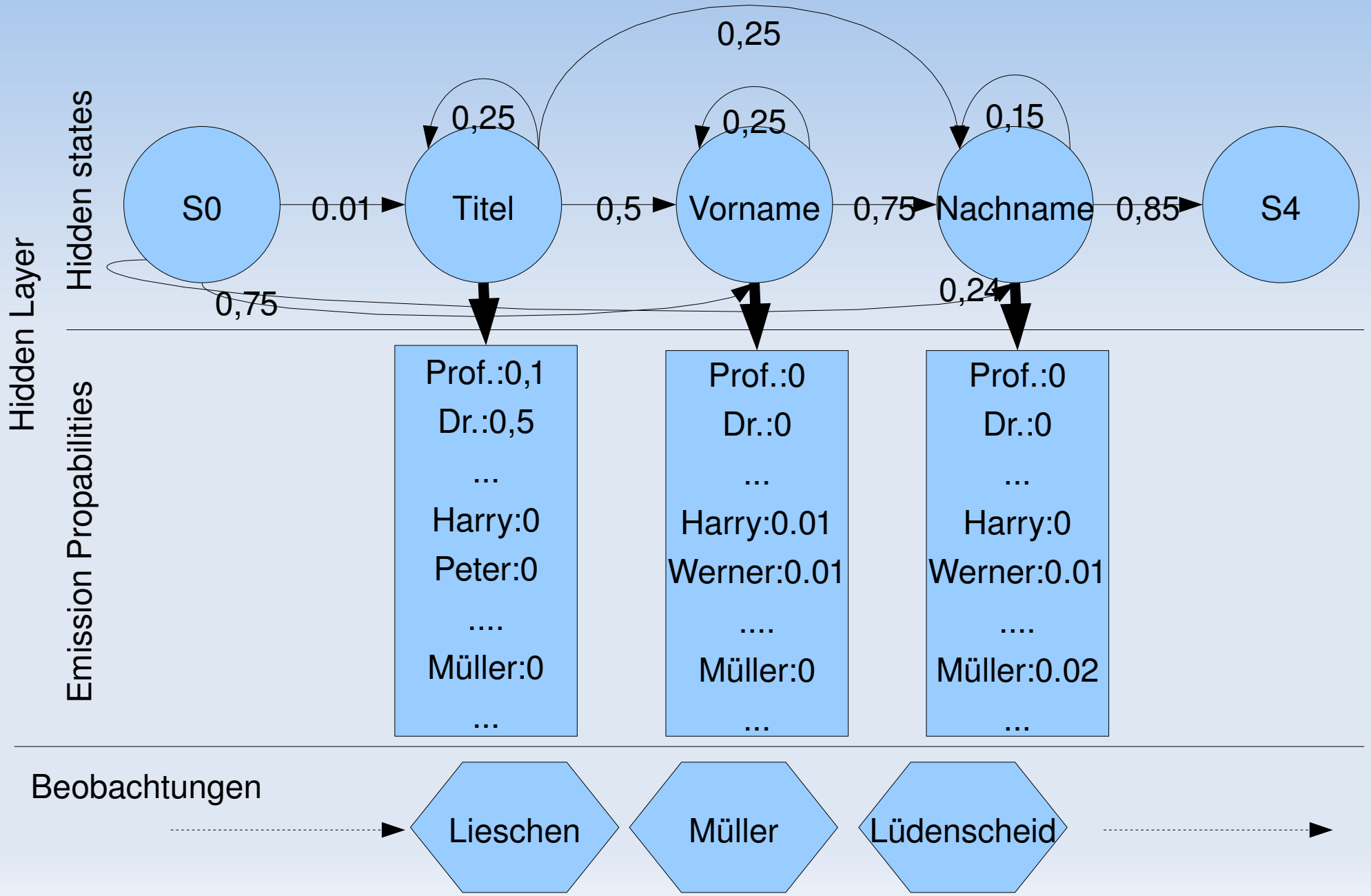
Naive Bayes (erg.)

- Hierarchy mit 51 Blättern
- 250(-40)/33.000 Dokumente als Trainingsmenge
  - Keywords = 66% (allerdings besondere Kenntnisse)
  - Bayes = 62%
  - Bayes + shrinkage = 70%
- Lässt sich mit unklassifizierter Trainingsmenge und Clustering noch steigern

Information Extraction

- Automatisches Auffinden besonderer Informationen in einem Text.
  - z.B. Wortarten
  - **hier Auffinden von Referenzen**
- Identifikation anhand des Aufbaus
  - HMM

Hidden Markov Models (Beispiel)



Hidden Markov Models (Formal)

Q : Menge der Zustände (hidden)

X : Menge der beobachtbaren Zustände

$$P(x|M) = \sum_{q_1,\ldots,q_l \in Q^l} \prod_{k=l}^{l+1} P(q_{k-1} \to q_k) P(q_k \uparrow x_k)$$

Wahrscheinlichkeit das x von dem HMM M erzeugt wurde

$$V(x|M) = \underset{q_1,\ldots,q_l \in Q^l}{argmax} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k) P(q_k \uparrow x_k)$$

Wahrscheinlichkeit des wahrscheinlichsten Pfades im HMM M
der die Ausgabe x erzeugt hat

Ziel: Ausgabe des Wahrscheinlichsten Pfades ...

-> Viterbi Algorithmus<-

Hidden Markov Models (VITERBI)

für alle beobachteten Zustände x:

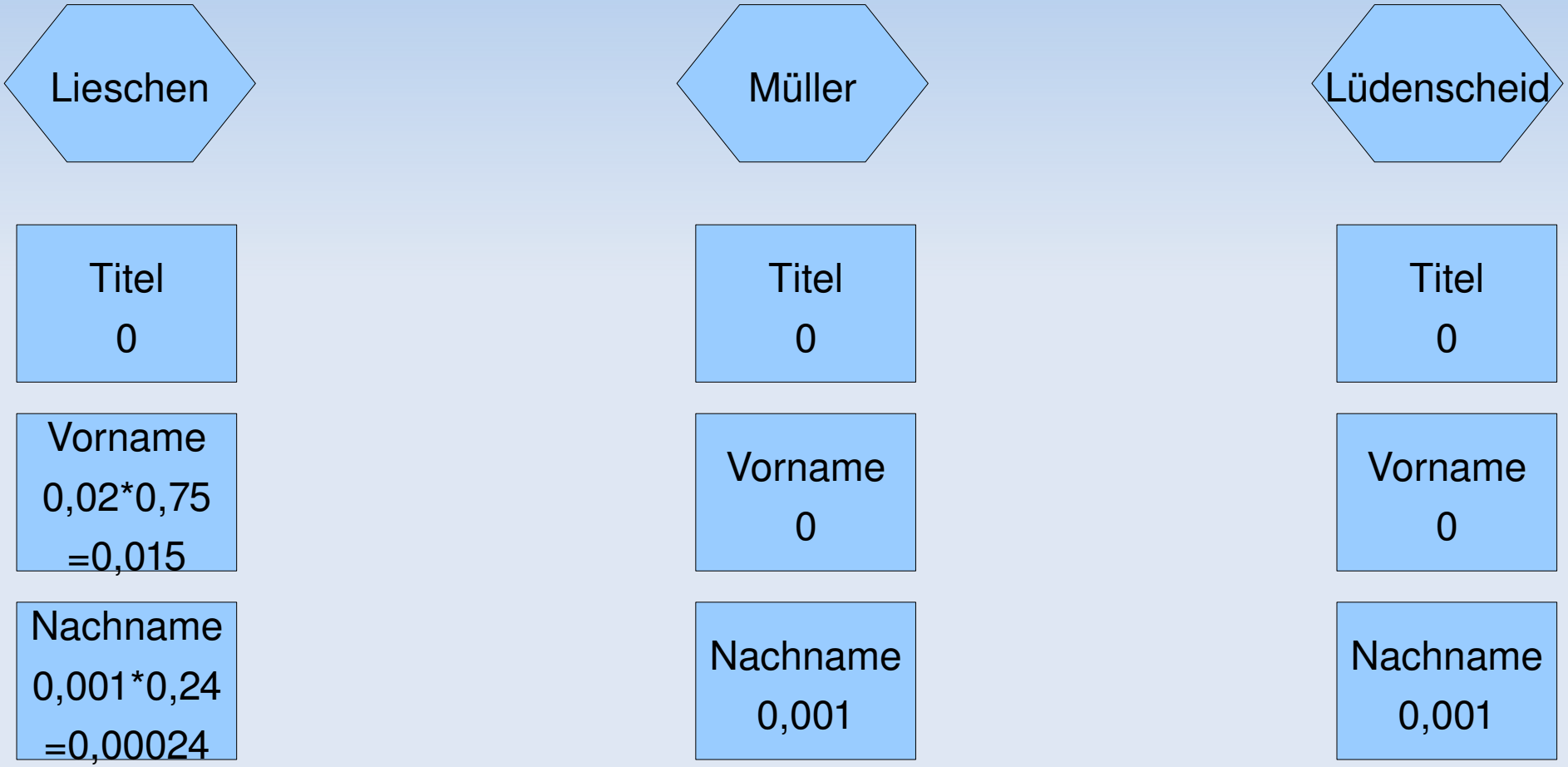
für alle hidden-Zustände (k)

für aller hidden Zustände (j)

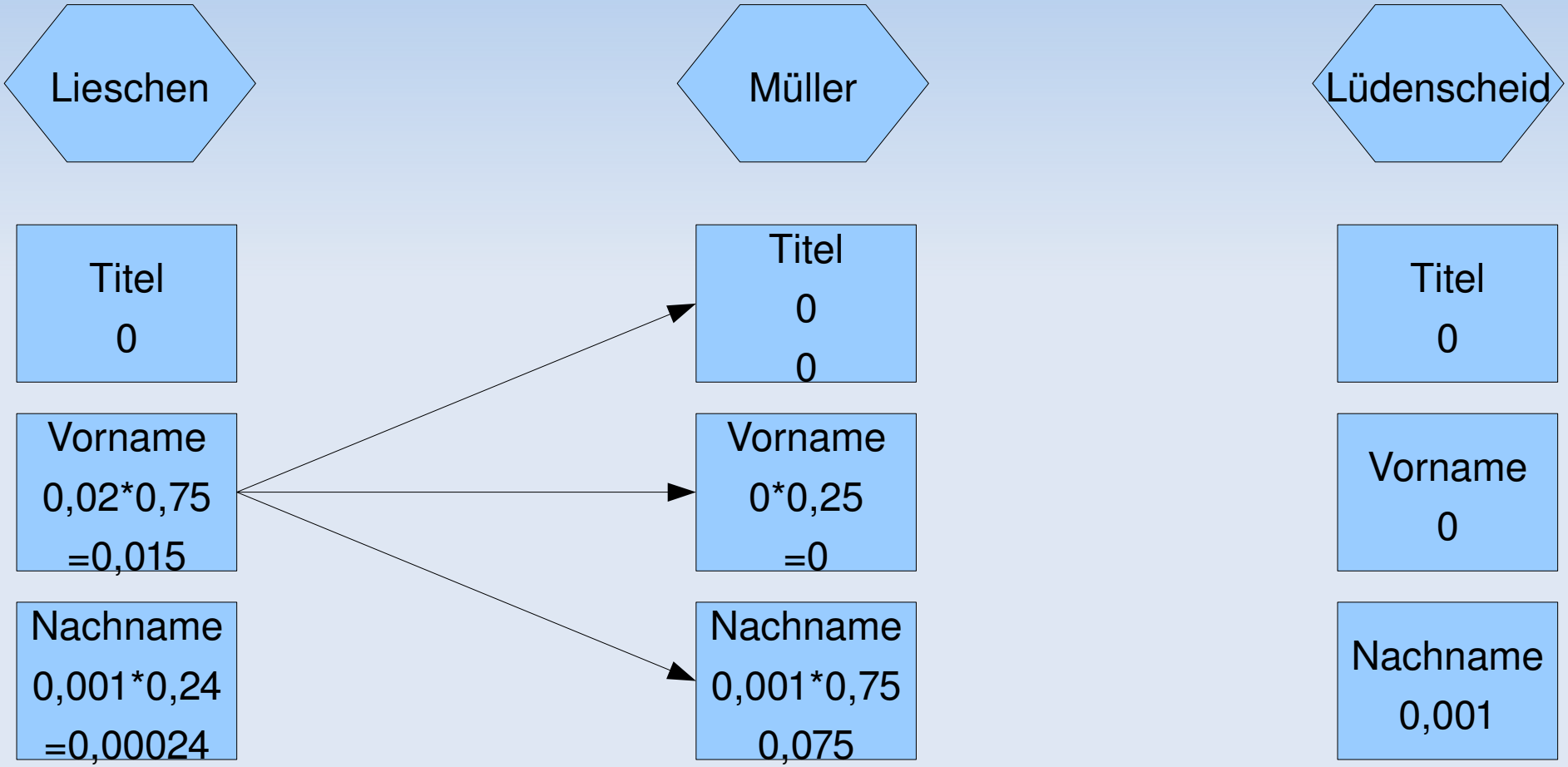
result = P(k->j)*P_SummePfade(k)*P(x|j)

(merke den Pfad und Maximum)

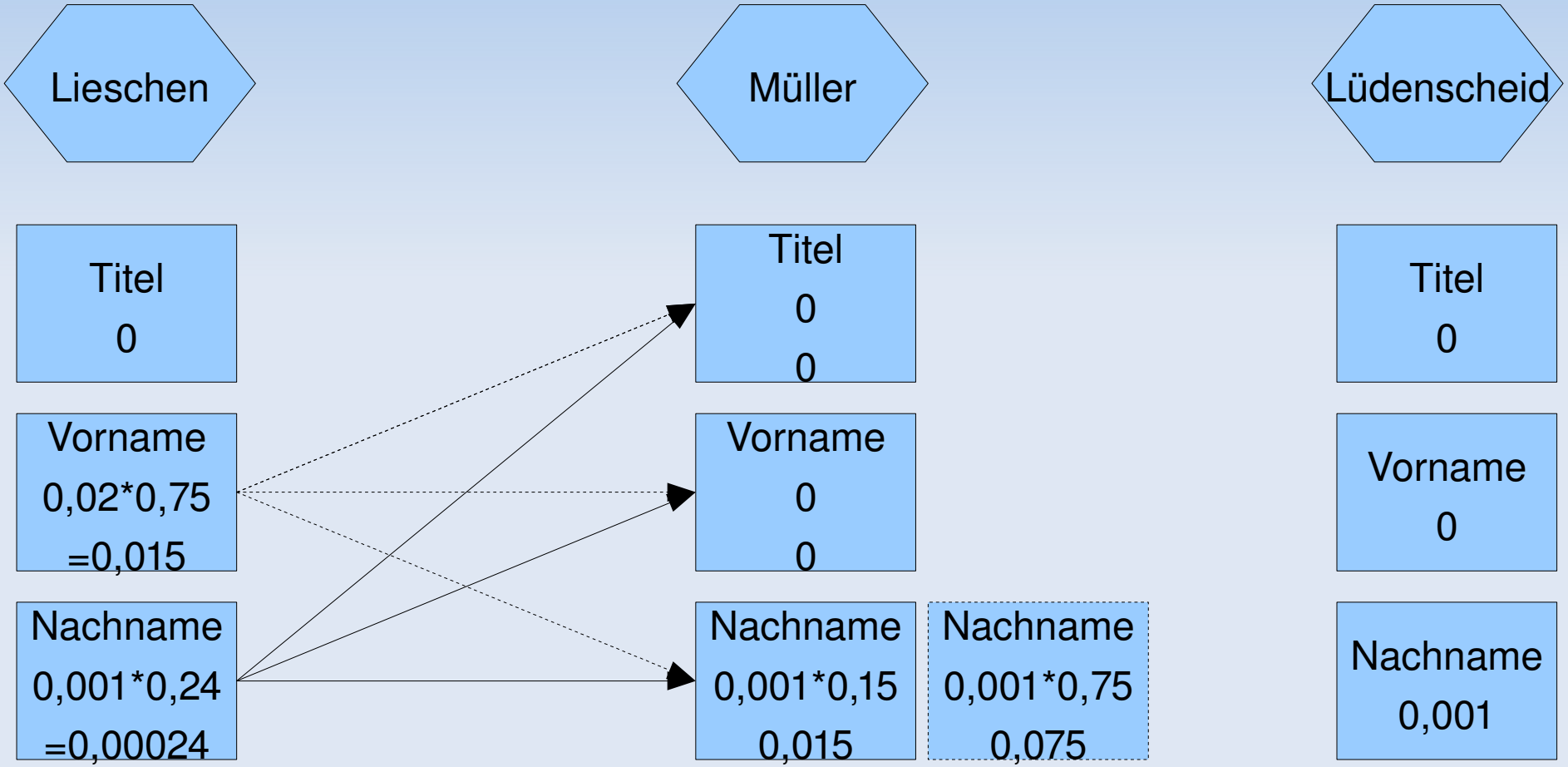
Hidden Markov Models (VITERBI)



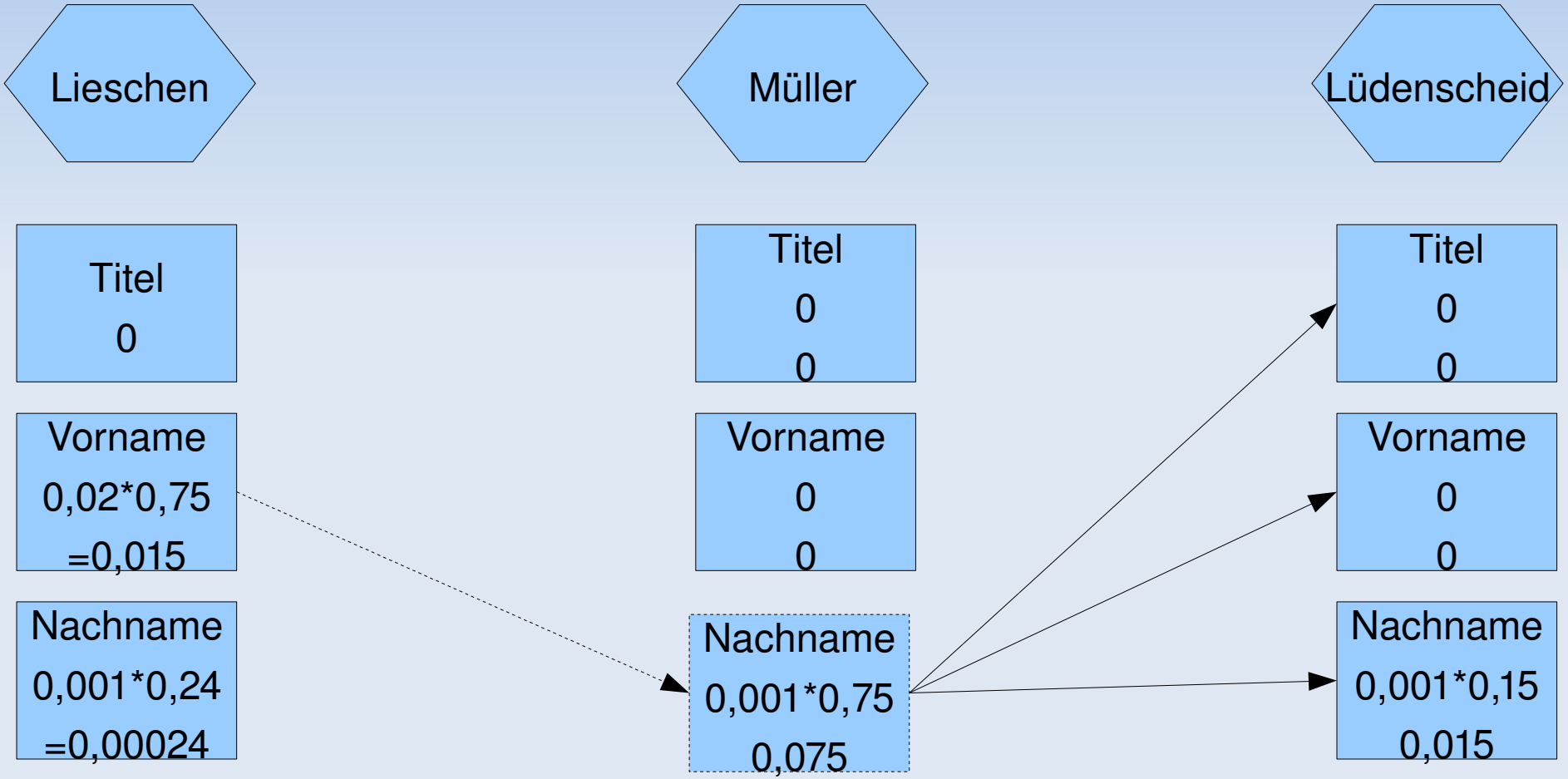
Hidden Markov Models (VITERBI)



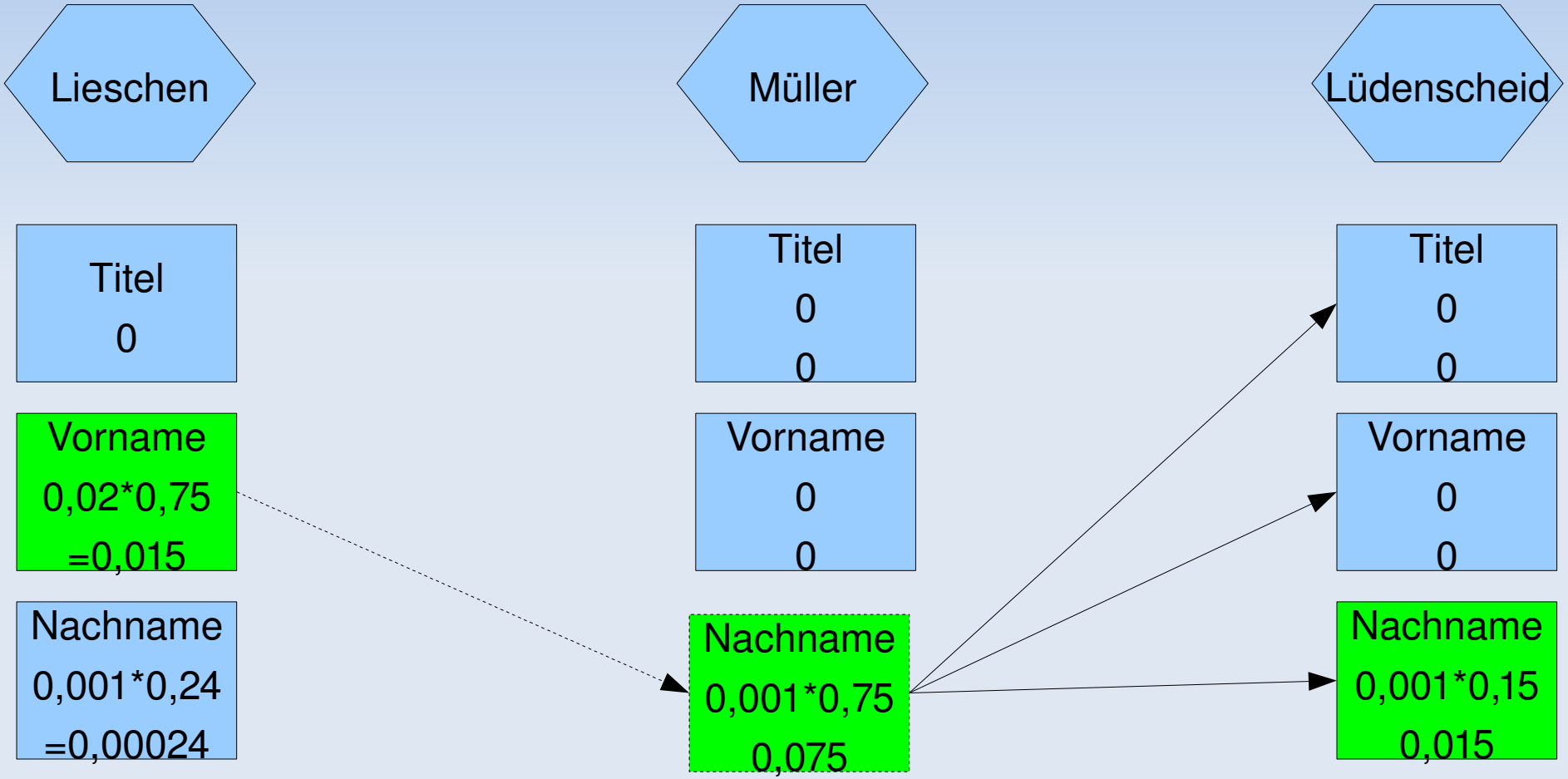
Hidden Markov Models (VITERBI)



Hidden Markov Models (VITERBI)



Hidden Markov Models (VITERBI)



$$V(\text{Lieschen Müller Lüdenscheid}|M) = 0{,}015 \cdot 0{,}075 \cdot 0{,}015 = 1{,}6875 \cdot 10^{-5}$$

Hidden Markov Models (erg.)

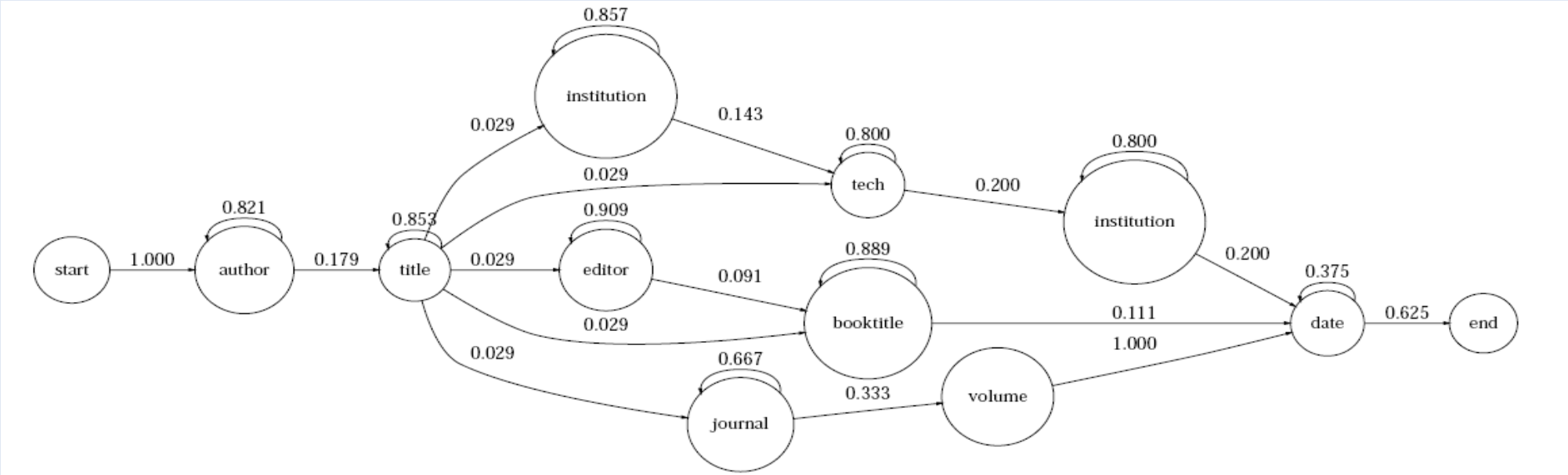
- Basis
  - 500 Referenzen aus 500 Dokumenten
  - 13 Klassen
    - Titel / Autor / Gesellschaft / Ort / Bemerkung / Editor / Verleger / Datum / Seitenzahl / Nummer / Zeitschrift / Buchtitel / Bericht

Hidden Markov Models (erg.)

- HMMs
  - HMM-0
    - Jede Klasse mit jeder anderen mit gleicher WK
  - HMM-1
    - Jede Klasse mit jeder anderen mit Häufigkeiten aus realen Daten
  - HMM-2
    - HMMs aus Dokumenten (Real)
    - Ein Startzustand mit gleicher WK für alle HMMs
    - Zusammenfassung gleicher Nachbarzustände

Hidden Markov Models (erg.)

- HMM-3
  - HMM-2
  - Klassen die gleiche eingehende / ausgehende Transitionen + gleiches Label haben werden zusammengefügt



(„Building Domain Specific Search Engines with Machine Leaning Techniques“, 1999 McCallum et al,, S. 10 (5.2 - Figure 5))

Hidden Markov Models (erg.)

Model	# states	Accuracy	
		Any word	Punc word
HMM-0	13	59.2	80.8
HMM-1	13	91.5	92.9
HMM-2	1677	90.2	91.1
HMM-3	46	91.7	92.9

Table 1: Word classification accuracy results (%) on 200 test references (4479 words).

(„Building Domain Specific Search Engines with Machine Leaning Techniques“, 1999 McCallum et al,, S. 11 (5.2 – Table 1))

Literaturtips / Links

- ReinforcementLearning
  - http://www.nbu.bg/cogs/events/2000/Readings/Petrov/rltutorial.pdf
  - http://www.informatik.uni-freiburg.de/~ml/teaching/ws04/lm/20050118_FocussedCrawling_Fischer.ppt
- Bayes
  - http://www.ke.informatik.tu-darmstadt.de/lehre/ss05/web-mining.html
    - http://www.ke.informatik.tu-darmstadt.de/lehre/ss05/web-mining/wm-tm.pdf
- HMM / Viterbi
  - http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html