Softwareprojektpraktikum Maschinelle Übersetzung

Matthias Huck, Daniel Stein
{huck,stein}@i6.informatik.rwth-aachen.de

Besprechung 4. Aufgabe 11. Juni 2010

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6
Computer Science Department
RWTH Aachen University, Germany

Outline

1 Wiederholung

$$\begin{aligned}\hat{e}_1^{\hat{I}} &= \arg\max_{e_1^I} \left\{ p(e_1^I | f_1^J) \right\} \\ &= \arg\max_{e_1^I} \left\{ p(e_1^I) \cdot p(f_1^J | e_1^I) \right\}\end{aligned}$$

Wo stehen wir?

Verlauf des Praktikums

1. Extraktion einer Wort-zu-Wort-Übersetzungstabelle (Alignment vorgegeben)
2. Implementieren eines Decoders, A*-Suche für n-best Listen
3. Berechnung von Fehlermaßen WER, PER und BLEU
4. Phrasenextraktion und Sprachmodell
5. Optimierung der Modellgewichte: Downhill-Simplex und Powell's
6. ???

2 IWSLT Evaluation 2010

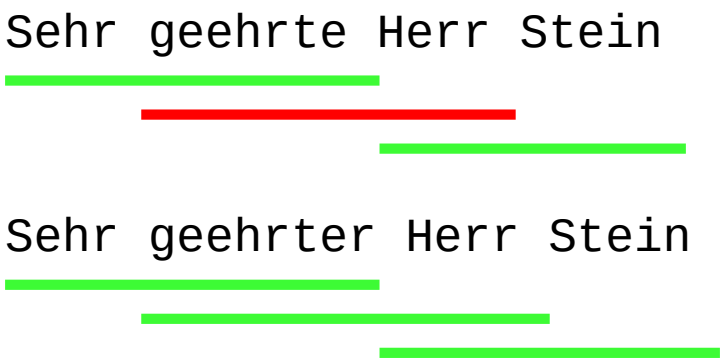
- International Workshop on Spoken Language Technology
- `http://iwslt2010.fbk.eu/`
- Evaluierungen:
  - TALK: English–French
  - DIALOG: Chinese–English
  - BTEC: Turkish, Arabic, French–English
- Trainingsdaten: 28 May 2010
- Testdaten: 23 August 2010
- Einsendung: 6 September 2010
- Ergebnisse: 12 September 2010
- MT System-Beschreibungen: 14 October 2010
- Konferenz: 2–3. Dezember, Paris

3 Sprachmodell

- **Problem: insbesondere an Phrasengrenzen können Fehler auftreten**
- **Beispiel:**
  - **Zu übersetzender Satz: Dear Mr. Stone**
  - **Phrasentabelle:**
    - **2 2 # Dear # Sehr geehrte**
    - **3 3 # Dear # Sehr geehrter**
    - **2 2 # Mr. Stone # Herr Stein**
  - **Ergebnis:**
    - **Sehr geehrte | Herr Stein (Kosten: 4)**
    - **Sehr geehrter | Herr Stein (Kosten: 5)**

Warum Sprach-Modell?

- **Bewertung durch ein Bigram Language Model:**



- **Das obere Beispiel wird größere Kosten verursachen (z.B. 5)**
- **Das untere Beispiel wird bessere Kosten verursachen (z.B. 2)**
- **Neues Ergebnis:**
  - **Sehr geehrter | Herr Stein (Kosten: 5+2)**
  - **Sehr geehrte | Herr Stein (Kosten: 4+5)**

Ergebnisse in der Spracherkennung

- **Allgemeine Formel:** $Pr(e_1^I) = \prod_{i=1}^{I} Pr(e_i|e_1^{i-1})$
- **In der Praxis Beschränkung auf** $n-1$ **letzte Wörter, die sogenannte history** h
- $h = e_{i-n+1}^{i-1}$**, typischerweise für** $n \in \{3, \ldots, 7\}$

- **Testergebnisse auf dem Wall Street Journal 5k Task:**

	Phoneme Error Rate [%]	Word Error Rate [%]
Ohne Sprachmodell	13.9	40.0
+ Sprachmodell: Unigram	8.4	22.9
Bigram	2.8	6.9
Trigram	1.9	4.5

Einfluss des Sprachmodells

Example from the Wall Street Journal 5k task:

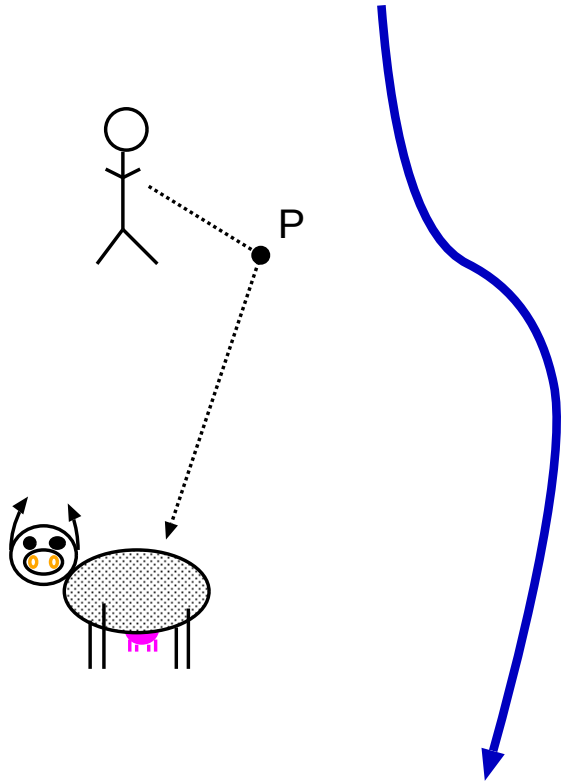
LM	recognized	errors
0-gram	h ih t s eh n uh t ur z n ih g oh sh ee ey t ih ng — — s ey l — — s ur t un aa s eh t s aw n t uh b r oh k ur ih j y ooh n ih t s	11
	HIT SENATORS — — NEGOTIATING — SALE — CERTAIN ASSETS ONTO — BROKERAGE UNIT'S	9
1-gram	ih t s s eh n ih t ih z n ih g oh sh ee ey t ih ng — — s ey l — — s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t	6
	ITS SENATE — IS NEGOTIATING — SALE — CERTAIN ASSETS OF THE BROKERAGE UNIT	5
2-gram	ih t s eh d ih t ih z n ih g oh sh ee ey t ih ng dh uh s ey l aw v s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t	0
	IT SAID IT IS NEGOTIATING THE SALE OF CERTAIN ASSETS OF THE BROKERAGE UNIT	0

Maximum Likelihood und Lagrange Multiplikator

- **Berechnung: Maximum Likelihood**
  - **Wahrscheinlichkeiten soll auf unseren Trainingsdaten maximal sein**
  - **Bestimmung der besten Wahrscheinlichkeitsparameter θ**
- **Gesucht:** $\arg\max_\theta \left\{ \prod_{i=1}^{I} p_\theta(e_i|h_i) \right\}$
- **Nebenbedingung:** $\sum_w p(w|h) = 1 \qquad \forall h$
- **Methode: Lagrange Multiplikator**

Lagrange Multiplikator

- **Aufgabe: Kuh melken**
- **Vorher: Eimer im Fluss waschen**
- **Gesucht: Kürzester Weg**
  $f(P) =$ **Weg**$(Ich, P) +$ **Weg**$(P, Elsa)$
- **Nebenbedingung: P ist im Fluss**
  $g(P) =$ **AbstandFluss**$(P) = 0$

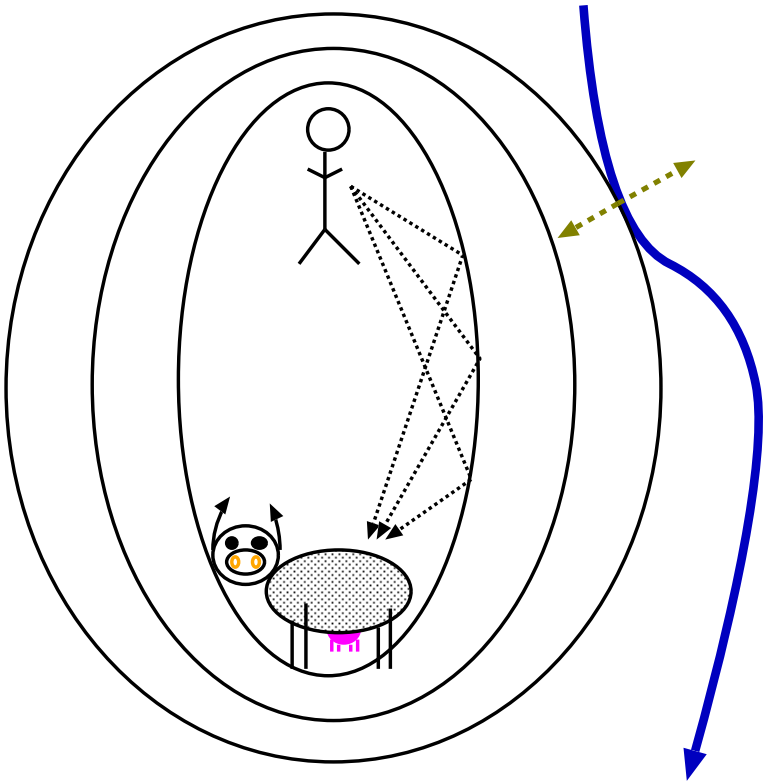


Lagrange Multiplier

- **Jede Ellipse: Gleiche Werte für $f(P)$**
- **Gesucht: kleinste tangentiale Ellipse**

$$\frac{\partial f}{\partial P} = \lambda \frac{\partial g}{\partial P}$$

- **Neue (Lagrange-)Funktion:**
  $F(P, \lambda) = f(P) + \lambda g(P)$
- **Setzen der Partiellen Ableitung gleich 0, danach lösen des DLG**



Konventionelle Berechnung des Sprachmodells

- **Rechnen im Logarithums:**
  - **Keine Änderung des Maximums (monoton für positive Werte)**
  - **Einfachere Handhabung**

$$\begin{aligned}
\arg\max_{\theta} \left\{ \prod_{i=1}^{I} p_\theta(e_i|h_i) \right\} &= \arg\max_{\theta} \left\{ \log \prod_{i=1}^{I} p_\theta(e_i|h_i) \right\} \\
&= \arg\max_{\theta} \left\{ \sum_{i=1}^{I} \log\ p_\theta(e_i|h_i) \right\} \\
&= \arg\max_{\theta} \left\{ \sum_{hw} N(h,w) \log\ p_\theta(w|h) \right\}
\end{aligned}$$

mit

$$N(h,w) := \sum_{i:(h,w)=(h_i,e_i)} 1$$

Konventionelle Berechnung des Sprachmodells

- **Lagrange Multiplikatoren:**

$$\tilde{F}(\{p_\theta(w|h); \lambda_h\}) = \sum_{hw} N(h,w) \log p_\theta(w|h) + \sum_h \lambda_h \left[\sum_w p_\theta(w|h) - 1 \right]$$

- **Partiellen Ableitungen:**

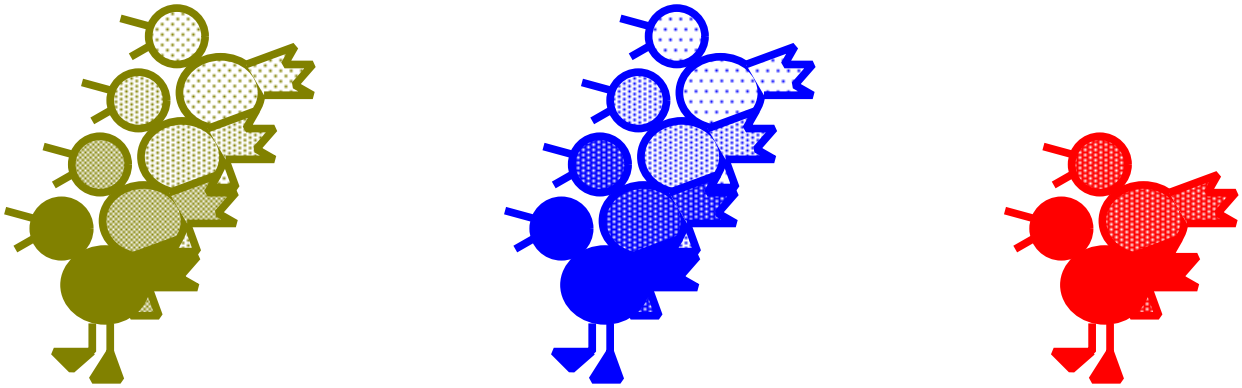
$$\frac{\partial \tilde{F}}{\partial p_\theta(w|h)} = \frac{N(h,w)}{p_\theta(w|h)} + \lambda_h = 0$$

$$\frac{\partial \tilde{F}}{\partial \lambda_h} = \sum_w p_\theta(w|h) - 1 = 0$$

- **Ergebnisse der Maximum Likelihood Schätzung:**

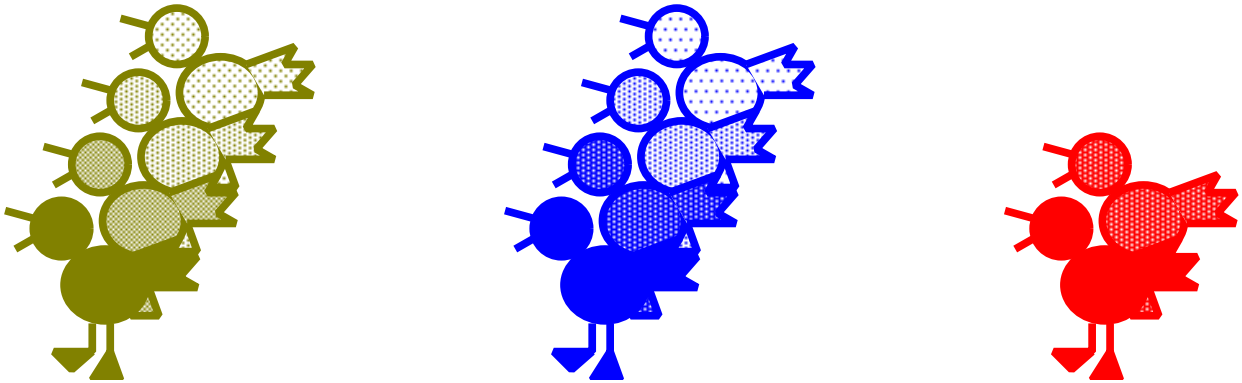
$$p(w|h) = \frac{N(h,w)}{\sum_{w'} N(h,w')} = \frac{N(h,w)}{N(h,\cdot)}$$

Spatz-Modell



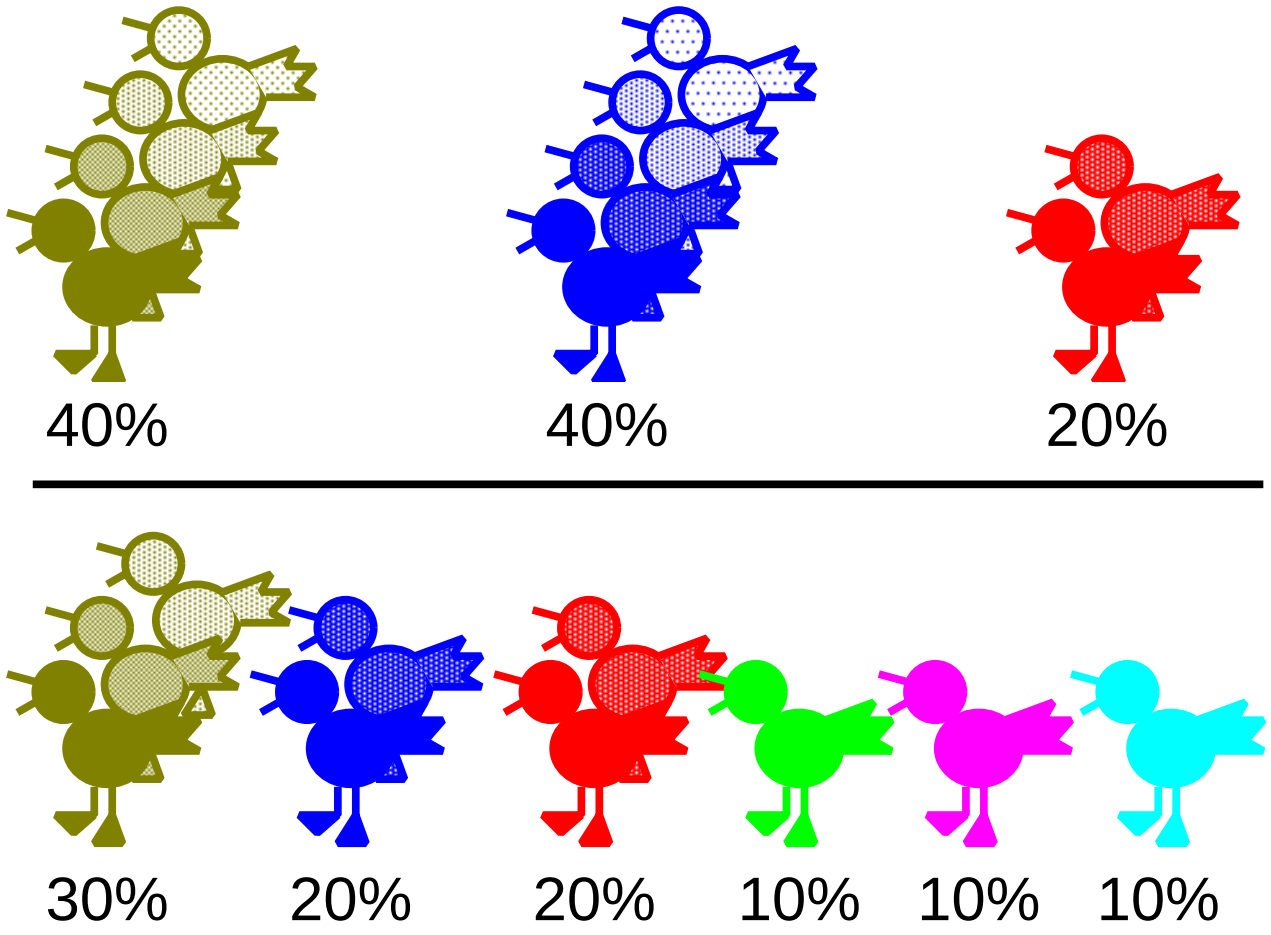
- **Wie wahrscheinlich ist ein Spatz der Farbe x?**

Spatz-Modell



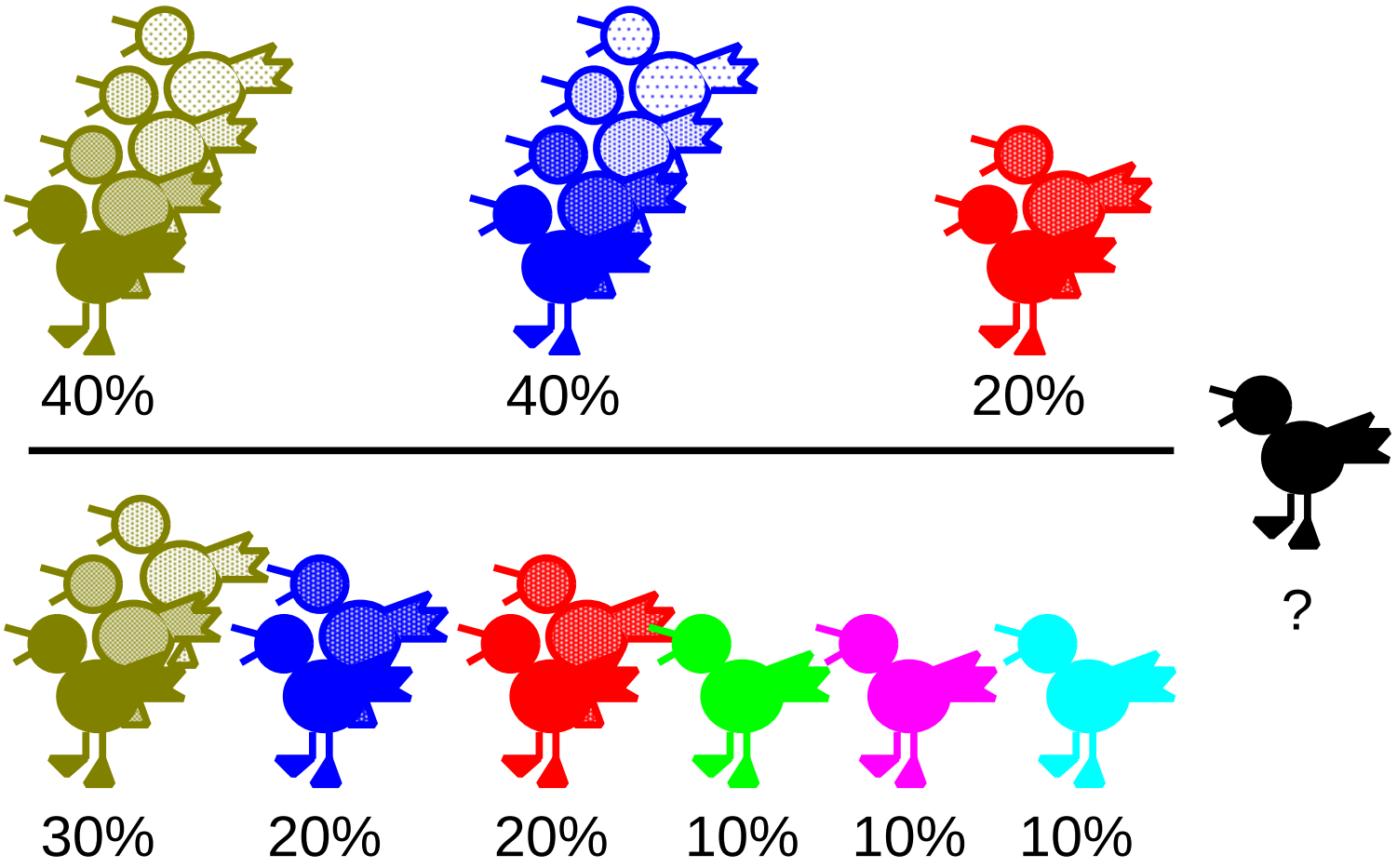
- **Konventioneller Ansatz: Relative Häufigkeit**

Spatz-Modell



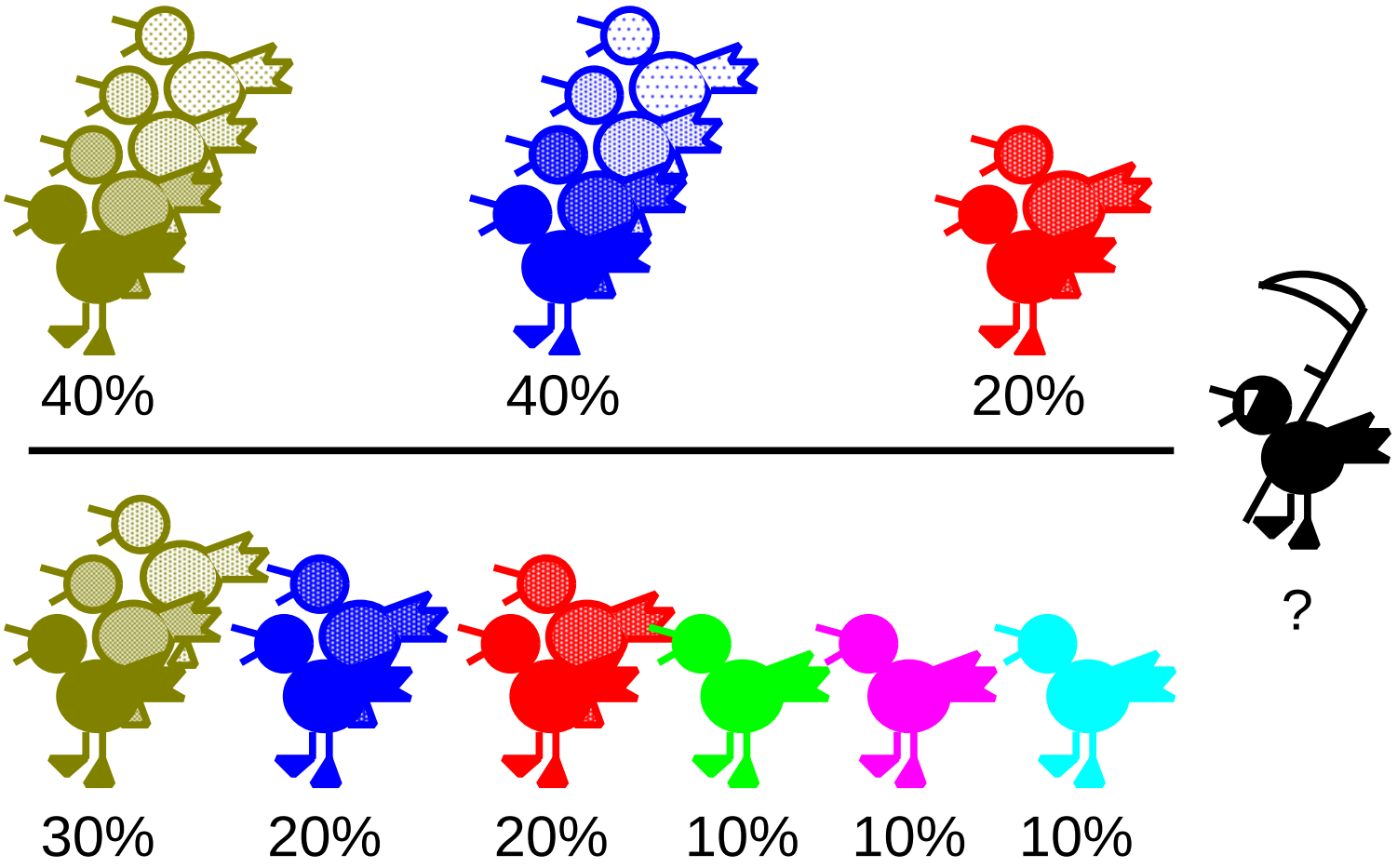
- **Welches von beiden Modellen ist stabiler?**

Spatz-Modell



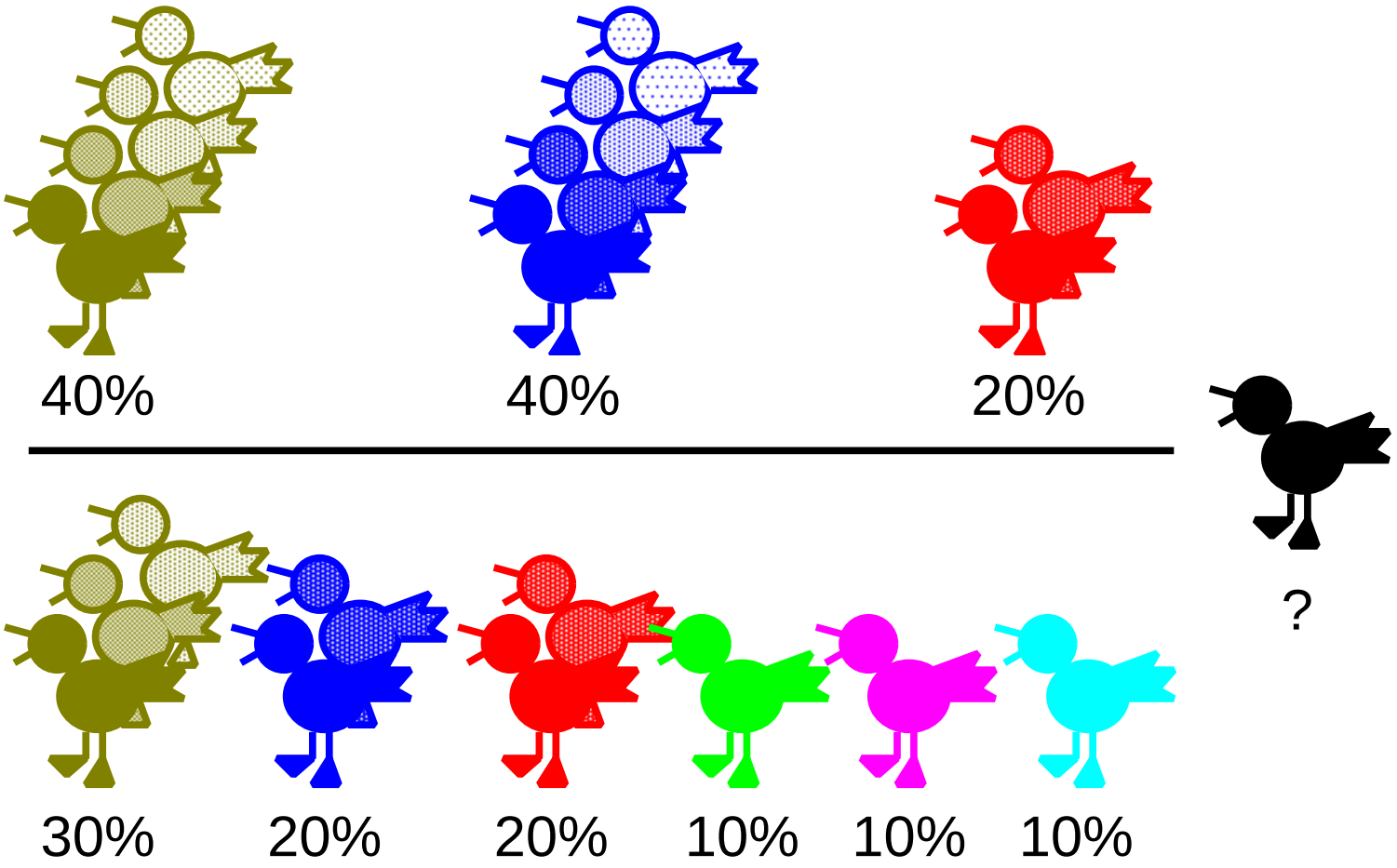
- **Was passiert mit ungesehenen Ereignissen? Wie wahrscheinlich sind sie?**

Spatz-Modell



- **Was passiert mit ungesehenen Ereignissen? Wie wahrscheinlich sind sie?**

Spatz-Modell



- **Was passiert mit ungesehenen Ereignissen? Wie wahrscheinlich sind sie?**

Spärliche Trainingsdaten für n-gram Sprachmodelle

- **Typisches Beispiel:**
  - **Anzahl der laufenden Wörter im Training:** $10 \cdot 10^6$
  - **Vokabulargröße:** $2 \cdot 10^4$
- **Art des Sprachmodells:**
  - **Bigram**
    Anzahl der möglichen Ereignisse: $|\text{Vokabular}|^2 = 4 \cdot 10^8$
    $\Rightarrow 10/400 = 2.5\%$ **können gesehen werden**
  - **Trigram**
    Anzahl der möglichen Ereignisse: $|\text{Vokabular}|^3 = 8 \cdot 10^{12}$
    $\Rightarrow 10/(8 \cdot 10^6) = 1.25 \cdot 10^{-4}\,\%$ **können gesehen werden**

Linear Discounting

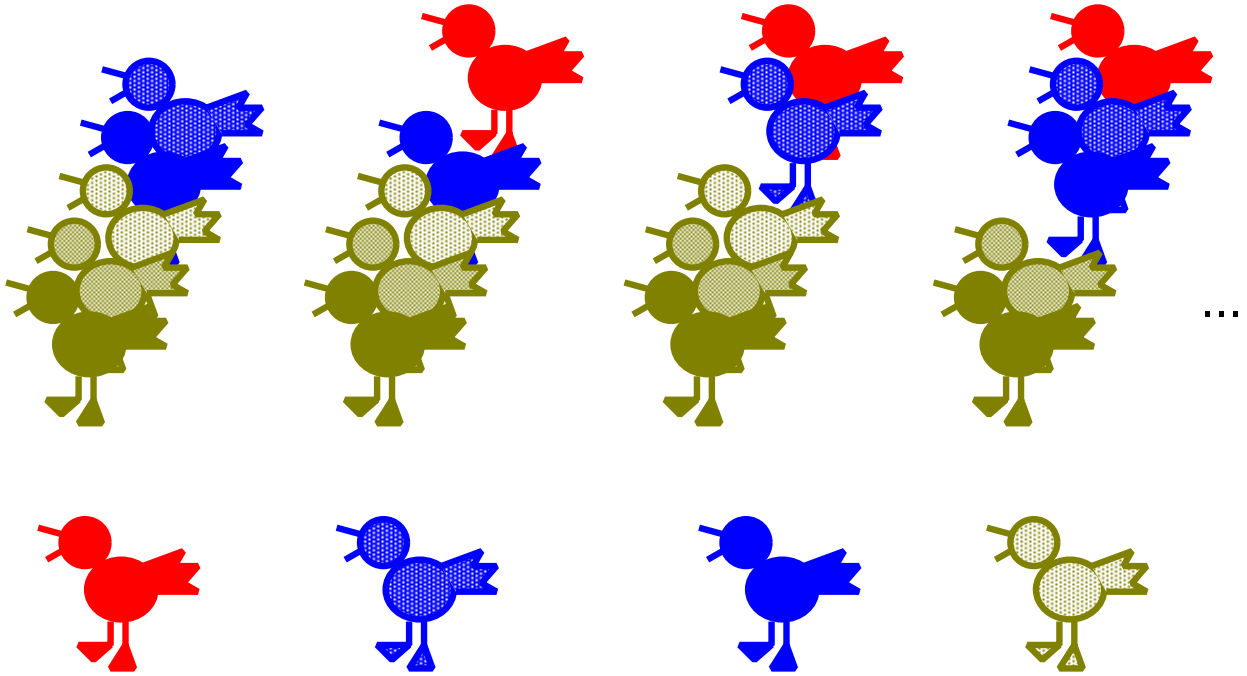
- **Discounting: Verschiebe Wahrscheinlichkeitsmasse zu ungesehen Ereignissen**
- **Linear: Proportional für jeden Count $N(h, w)$**

$$p(w|h) = \begin{cases} (1 - \mu_h) \cdot \dfrac{N(h,w)}{N(h,\cdot)} & \text{für } N(h,w) > 0 \\ \mu_h \cdot \dfrac{\beta(w|\overline{h})}{\sum\limits_{w': N(h,w')=0} \beta(w'|\overline{h})} & \text{für } N(h,w) = 0 \end{cases}$$

- μ_h: **komplette Wahrscheinlichkeitsmasse für ungesehene Ereignisse:**
- $\beta(w|\overline{h})$: **Renormalisierung für zweite (Backoff-)Wahrscheinlichkeit**
- μ_h **und** $\beta(w|\overline{h})$ **müssen ebenfalls aus den Trainingsdaten gelernt werden**

Leaving-One-Out für spärliche Trainingsdaten

- Generelle Idee: Ungesehene Ereignisse simulieren, indem man Trainingsdaten in zwei Teile aufteilt
- Leaving-One-Out: Jede einzelne Beobachtung gilt als Testbeobachtung
- die übrigen Beobachtungen dienen zum Training
- Sprachmodell: jeweils ein Ereignis $(h_i, e_i),\ i = 1, \ldots, I$ wird zurückbehalten



Parameterabschätzung mit Leaving-One-Out

- **Log-Likelihood Funktion:**

$$
\begin{aligned}
& F(\{\mu_h\}, \{\beta(w|\overline{h})\}) \; = \\
& = \sum_{hw: N(h,w)>1} N(h,w) \cdot \left[\log\,(1-\mu_h) \; + \; \log \frac{N(h,w)-1}{N(h,\cdot)-1} \right] \\
& \quad + \sum_{hw: N(h,w)=1} 1 \cdot \left[\log\; \mu_h \; + \; \log \frac{\beta(w|\overline{h})}{\sum_{w'} \beta(w'|\overline{h})} \right] \\
& = \ldots \\
& = F(\{\mu_h\}) \; + \; F(\{\beta(w|\overline{h})\}) \quad + \; \mathrm{const}(\{\mu_h\}, \{\beta(w|\overline{h})\})
\end{aligned}
$$

- **Ergebnis: Unabhängige Optimierung der beiden Funktionen**

Parameterschätzungen für Lineares Discounting

- Sei $N_1(h, \cdot)$ die Anzahl aller einmal gesehenen Ereignisse
- Discounting Parameter:
  - History-abhängig: $\mu_h = \frac{N_1(h, \cdot)}{N(h, \cdot)}$
  - History-unabhängig: $\mu = \frac{N_1(\cdot, \cdot)}{N(\cdot, \cdot)}$
- $\beta(w|\overline{h})$:
  - Language-Modell mit kleinerer history $\overline{h}$

4 Aufgabenblatt 4

- **Erweiterung der Übersetzung auf Phrasen**
- **Rescoring mit Language Model auf den n-Best Listen**

5 Sprachmodell: SRI Toolkit

- SRI Language Modelling Toolkit frei verfügbar für nicht gewerbliche Zwecke unter

  `http://www.speech.sri.com/projects/srilm/`).

- trainieren mit Kneser-Ney Smoothing

```
ngram-count -order 2 -lm e.lm.gz -text e
            -kndiscount1 -kndiscount2
```

Aufbau SRI-Language Model

ARPA LM Format:

```
\data\
ngram 1=n1
ngram 2=n2
...
ngram N=nN

\1-grams:
p    w          [bow]
...

\2-grams:
p    w1 w2      [bow]
...

\N-grams:
p    w1 ... wN
...

\end\
```

Ansteuern des SRI Language Models

Einbinden der SRI Language Model Bibliothek

- Einbinden von <Ngram.h> und <Vocab.h> im Programmcode.
- Kompilieren: Pfad der SRI Header Dateien (include/) angeben
  - -I*PATH_TO_HEADER*
- Kompilieren: Library includen mit den Optionen
  - -loolm -ldstruct -lflm -lmisc
  - -L*PATH_TO_LIBRARY*
- Anlegen der Klassen für
  - Vocabulars (Vocab)
    kann statt der alten Lexikon Klasse genutzt werden
  - Language Model (Ngram)

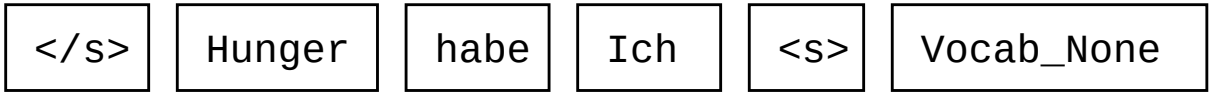
Die Vocab Klasse

- Lexikon für das Language Model
- die wichtigsten Befehle:

Vocab()	Konstruktor
string getWord(unsigned index)	gibt den String für index
unsigned addWord(char* Word)	fügt ein (falls unbekannt) und gibt den index zurück
int numWords()	Anzahl der Wörter
ssIndex()	Satzanfangszeichen
seIndex()	Satzendzeichen
unkIndex()	Index für das unbekannte Wort

Der Buffer

- vom Typ *VocabIndex
- muß initialisiert werden, z.B.
  `vocabBuffer(new VocabIndex[50])`
- evtl. muß auch die Größe neu angepasst werden (mit realloc)
- Satzformat:
  - Satzanfangs- und Satzendmarker (s. vorherige Folie)
  - umgekehrte Wortreihenfolge
  - mit Vocab_None abschließen
- Beispiel: "“Ich habe Hunger"” wird zu



- Berechnung: vom Satzendmarker (position 0) bis einschließlich dem ersten Wort
- also fürs obige Beispiel 4 Anfragen mit jeweiligem Kontext: </s>, Hunger, habe, Ich

Die Ngram Klasse

- Funktionalität für das Language Model
- die wichtigsten Befehle:

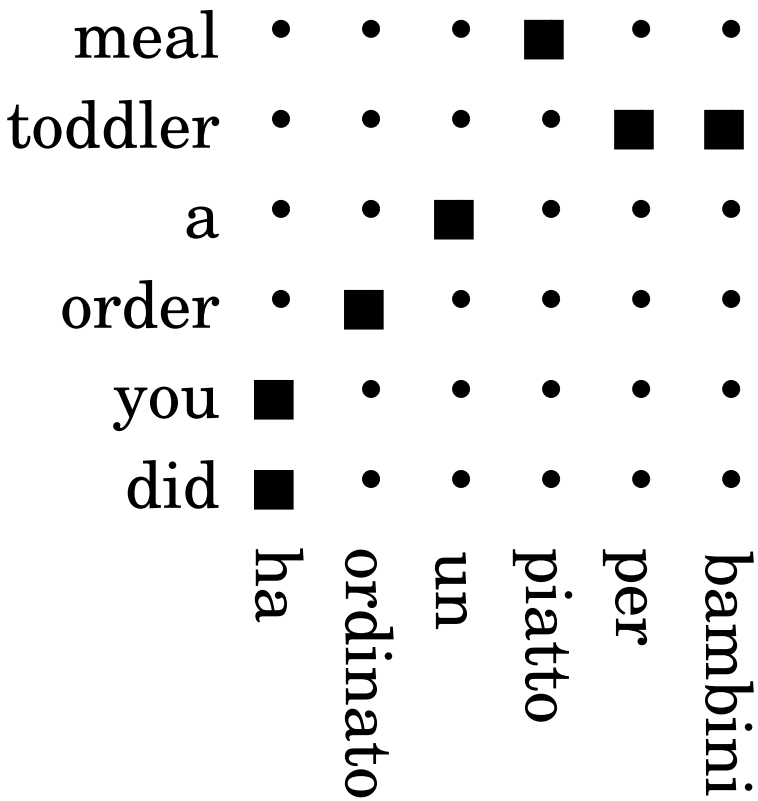
Ngram(Vocab *vocabulary, int lmOrder)	Konstruktor
read(File, boolean expandVocabulary)	liest ein Language Model ein
double wordProb(buf[pos], &buf[pos+1])	bewertet ein Wort an Position pos mit seinem Kontext Rückgabe: log-Score (negativ)

6 Extraktion: Gültige Phrasen

- Gegeben: ein Quellsatz f_i^J, ein Zielsatz e_1^I und ein zugehöriges Alignment A.
- Eine Phrasenpaar $(f_{j1}^{j2}, e_{i1}^{i2})$ wird als gültig angesehen, wenn
  - es mindestens ein Alignment zwischen den Phrasen gibt
  - alle Alignments nur innerhalb der Phrasen liegen, und
  - keines links, rechts, oben oder unten außerhalb

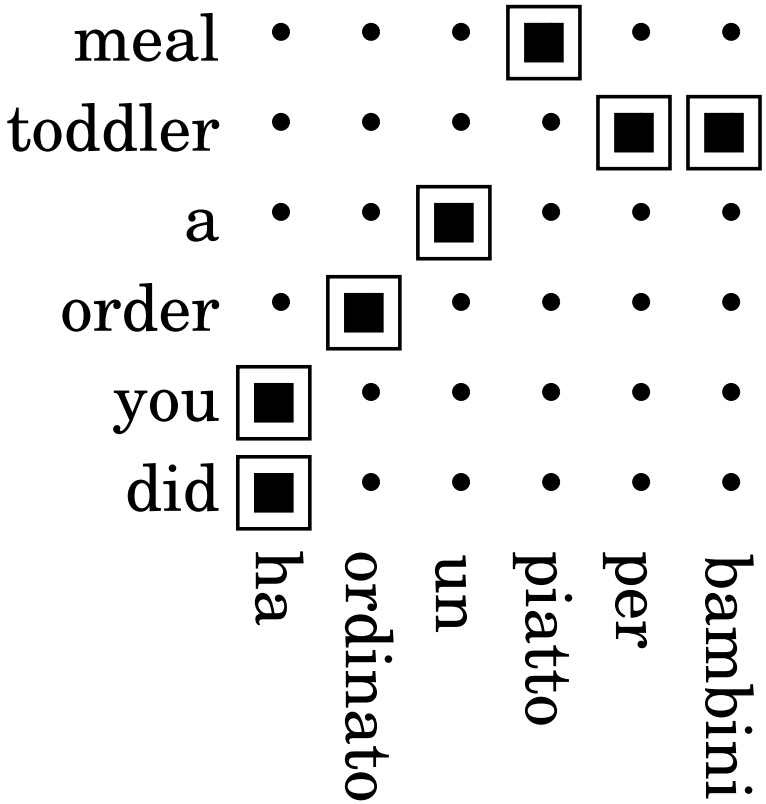
Beispiel

- **Beispiel-Alignment für das Sprachpaar Italienisch-Englisch (IWSLT)**



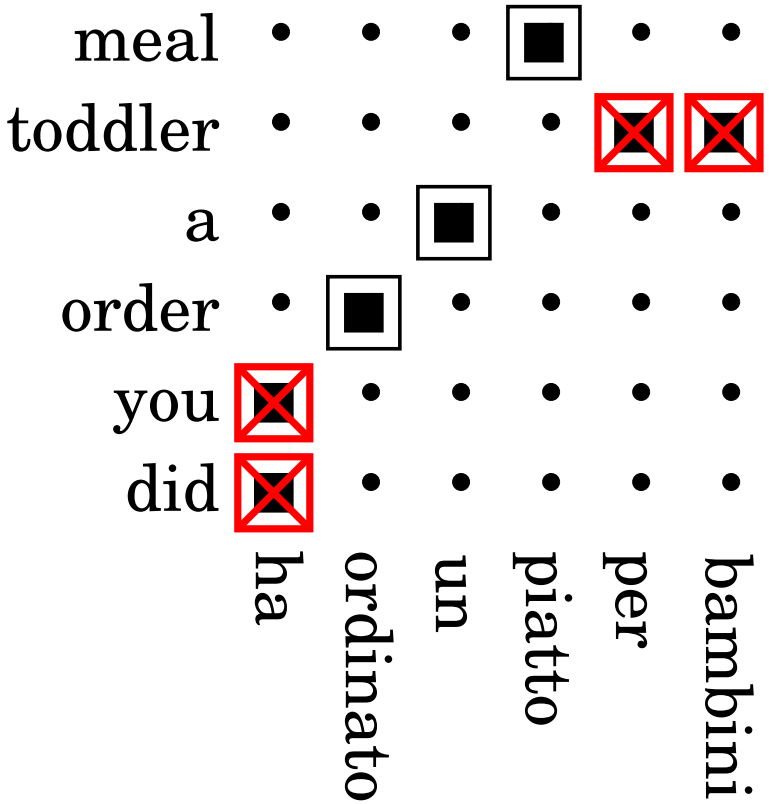
Beispiel

- **Single-Word Paare wie in Übung 1**



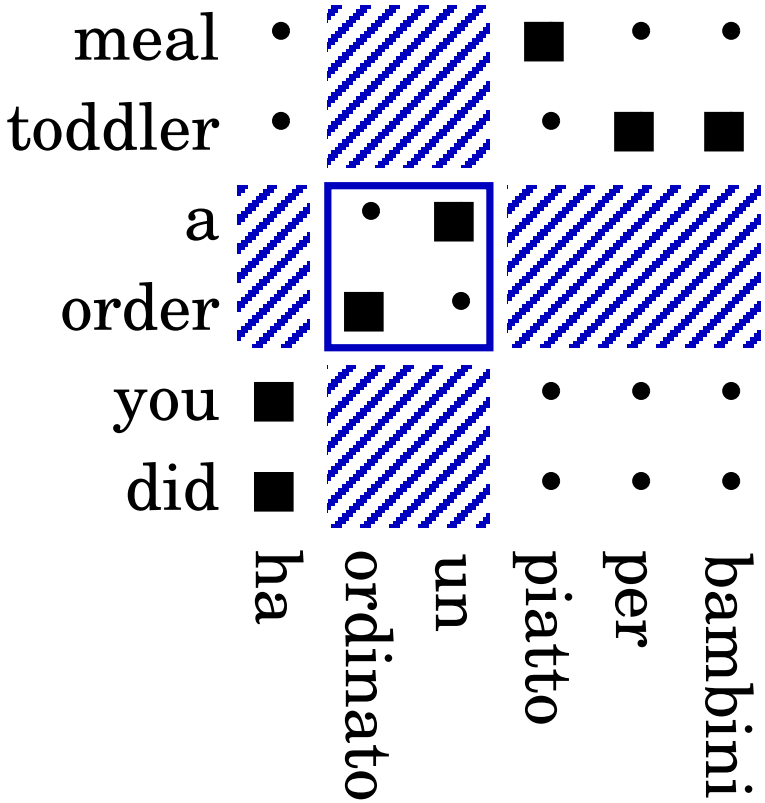
Beispiel

- **Jetzt ungültige Single-Word Paare**



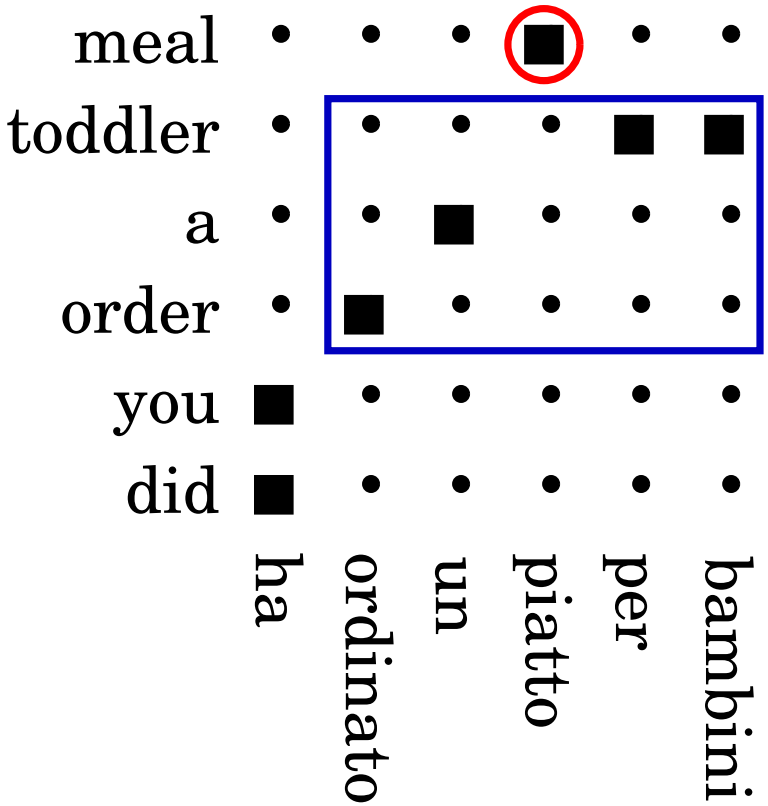
Beispiel

- **Gültiges Phrasenpaar**



Beispiel

- **Ungültiges Phrasenpaar**



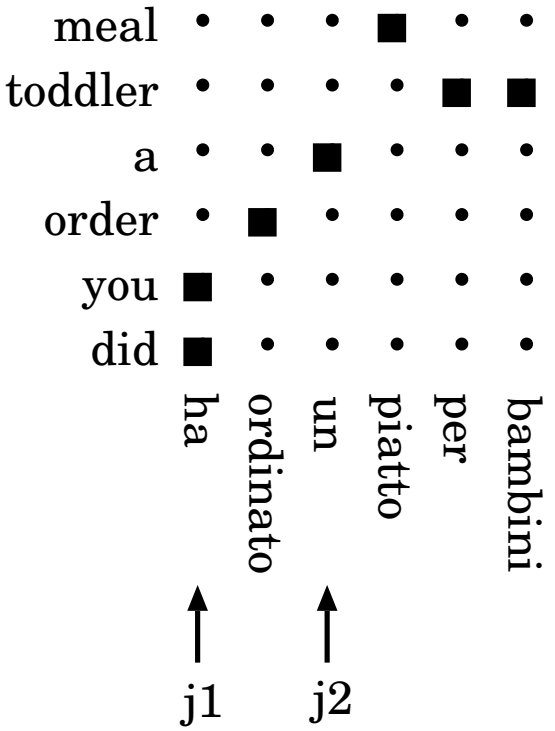
7 Extraktion: Implementierung

- **Verschachtelte Schleife (j1, j2) über den Quellsatz**
- **Ermitteln des minimalen (i1) und maximalen (i2) Wortes auf Zielseite**
- **Überprüfen, ob i1 und i2 ihrerseits nicht über j1 und j2 hinausgehen**

```
for j1 := 0 to J-1
  for j2 := j1 to J-1
    i1 = getMinZielAlignment(j1, j2)
    i2 = getMaxZielAlignment(j1, j2)
    if (getMinQuellAlignment(i1, i2) == j1 &&
        getMaxQuellAlignment(i1, i2) == j2)
      outputGueltigePhrase(j1, j2, i1, i2)
```

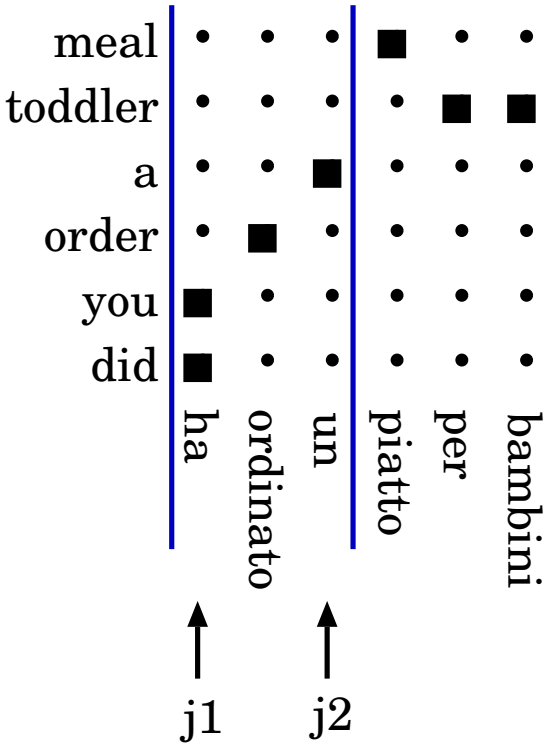
Beispiel

- **Beispiel für $j_1 = 0$ und $j_2 = 2$**



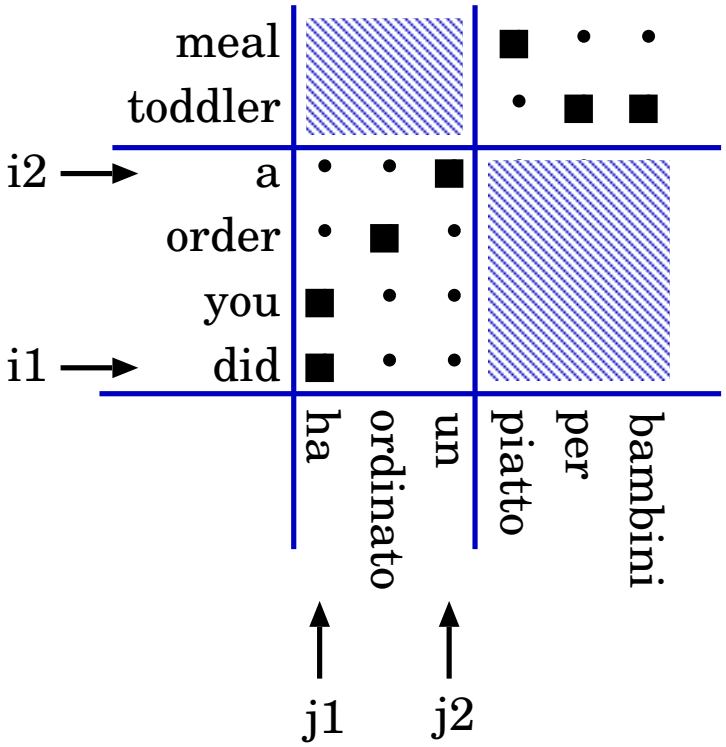
Beispiel

- **Beispiel für $j_1 = 0$ und $j_2 = 2$**
- **Ermitteln von i_1 und i_2**



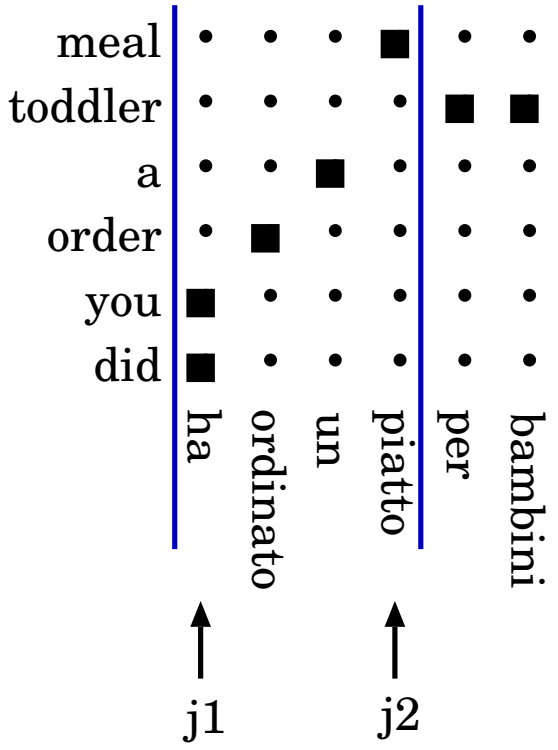
Beispiel

- Beispiel für $j_1 = 0$ und $j_2 = 2$
- Ermitteln von i_1 und i_2
- Gültige Phrase: ha ordinato un # did you order a



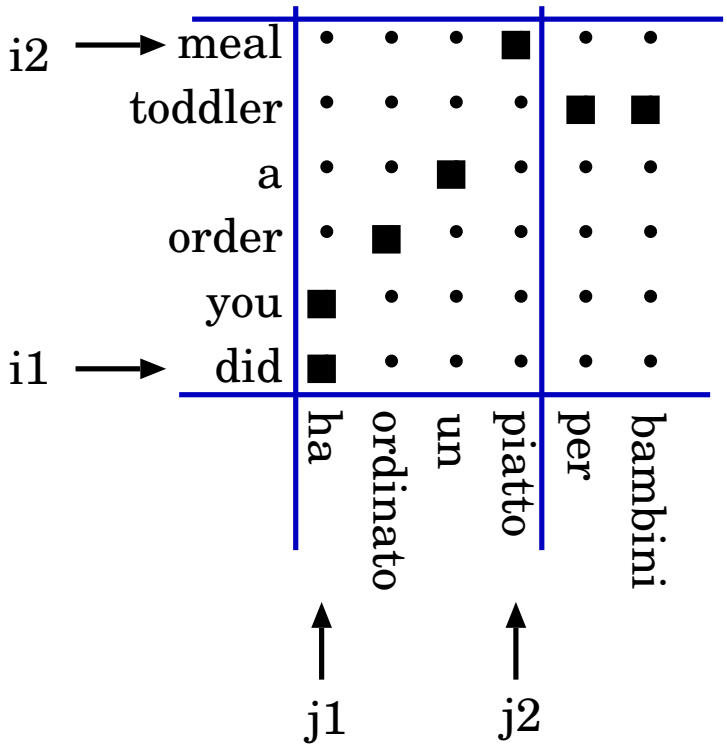
Beispiel

- **Nächste Iteration:**
  $j_1 = 0$ **und** $j_2 = j_2 + 1 = 3$



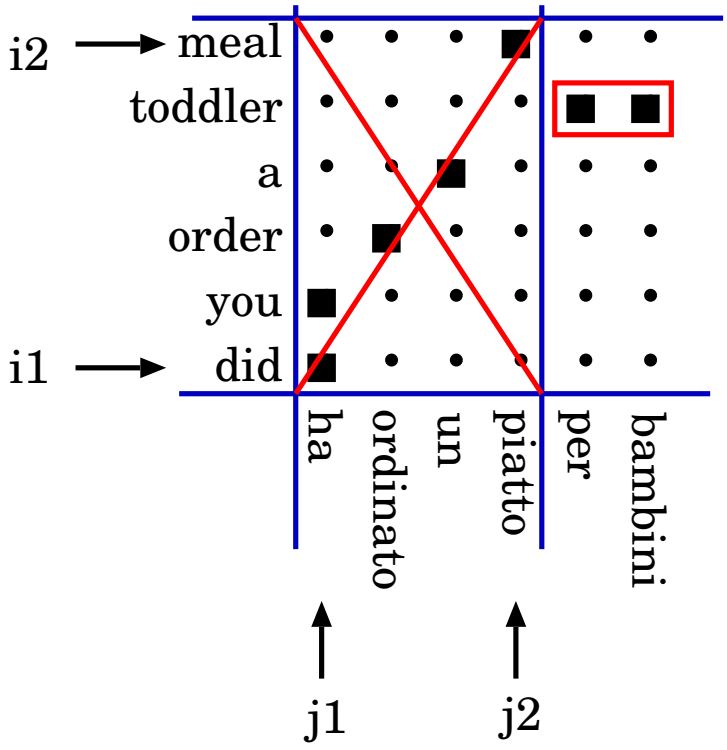
Beispiel

- **Nächste Iteration:**
  $j_1 = 0$ **und** $j_2 = j_2 + 1 = 3$
- **Ermitteln von** i_1 **und** i_2



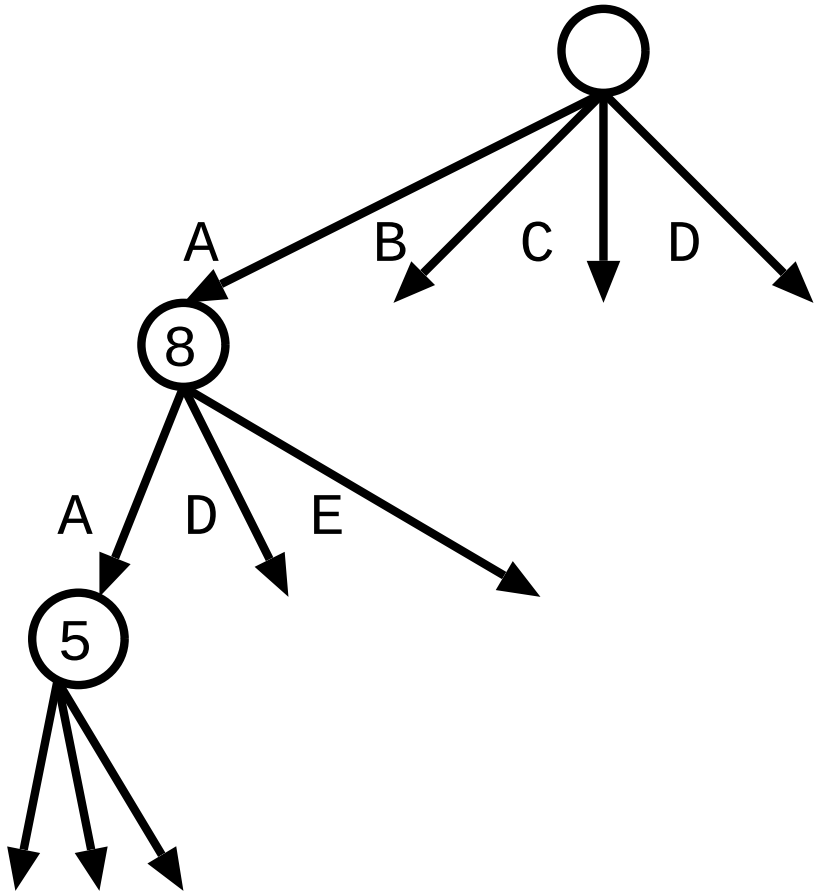
Beispiel

- **Nächste Iteration:**
  $j_1 = 0$ **und** $j_2 = j_2 + 1 = 3$
- **Ermitteln von** i_1 **und** i_2
- **Ungültige Phrase:**
  ha ordinato un piatto # did you order a toddler meal



Präfixbaum

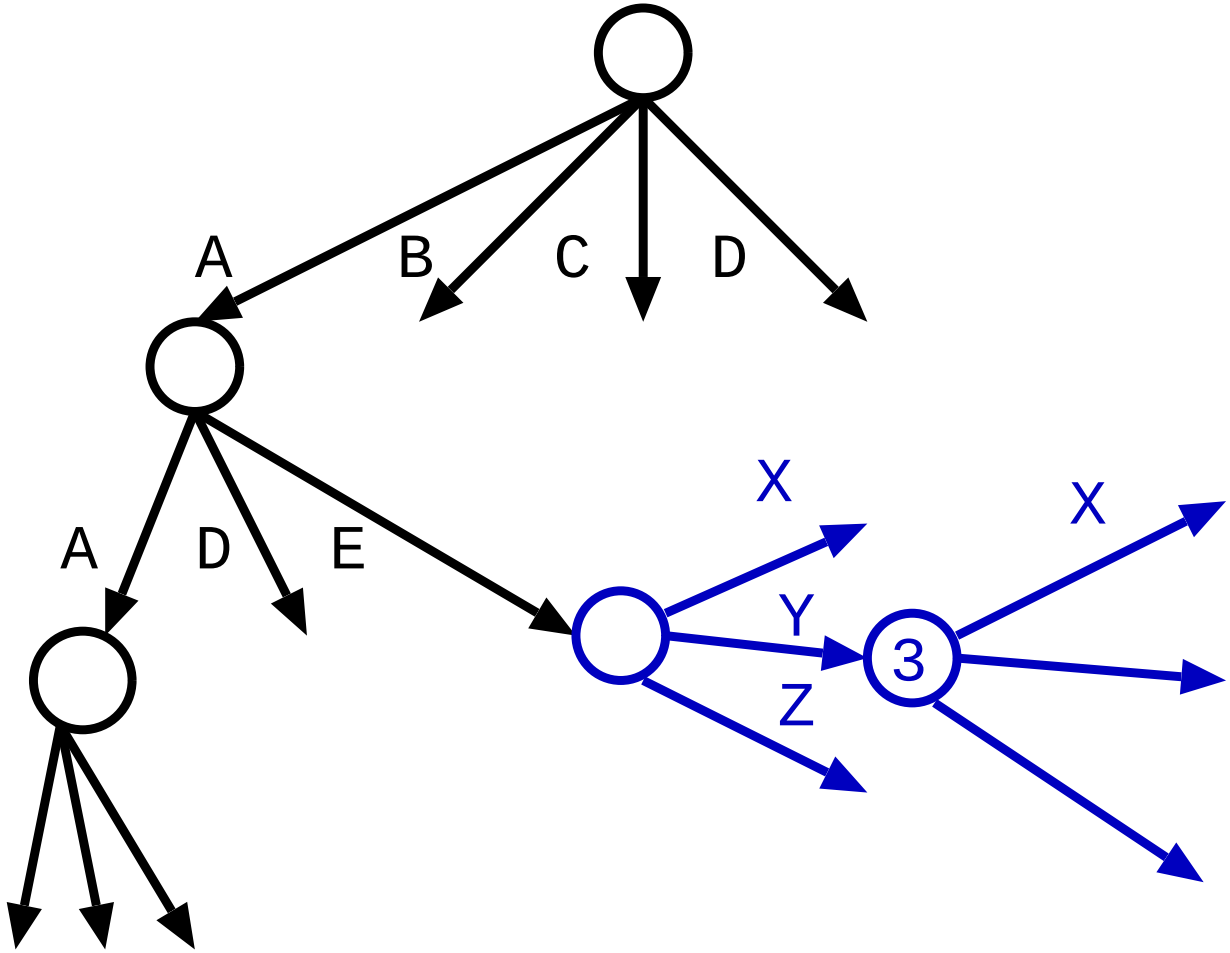
- Die Counts der Phrasen werden in Präfixbäumen abgespeichert



- Die Phrase A-A wurde 5 mal gesehen

Präfixbaum von Präfixbäumen

- **Die Counts der Phrasenpaare werden in Präfixbäumen von Präfixbäumen abgespeichert**



- **Das Phrasenpaar A-E # Y wurde 3 mal gesehen**

Erweiterung des Decoders und der A*-Suche auf Phrasen

Wenn ihr alles richtig gemacht habt, ist das trivial :-)

8 Fragen

Wir müssen unbedingt Raum für Zweifel lassen, sonst gibt es keinen Fortschritt, kein Dazulernen. Man kann nichts Neues herausfinden, wenn man nicht vorher eine Frage stellt. Und um zu fragen, bedarf es des Zweifelns.

— Richard P. Feynman