Softwareprojektpraktikum
Maschinelle Übersetzung

Matthias Huck, Markus Freitag
{huck,freitag}@i6.informatik.rwth-aachen.de

Vorbesprechung 6. Aufgabe 30. Juni 2011

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

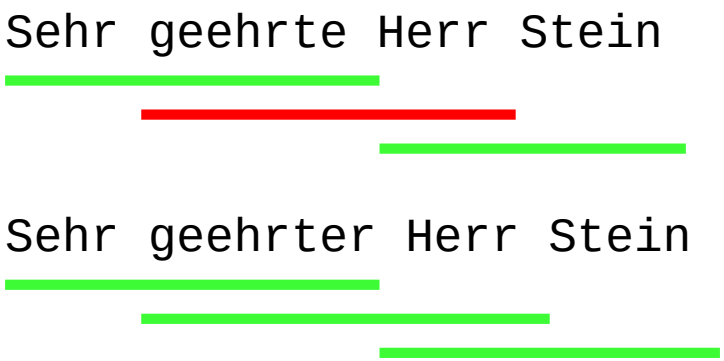
Outline

1 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 Sprachmodell?

- **Bewertung durch ein Bigramm-Sprachmodell:**



- **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: 1-gram	8.4	22.9
2-gram	2.8	6.9
3-gram	1.9	4.5

Einfluss des Sprachmodells in der Spracherkennung

Beispiel aus dem 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

Ergebnisse in der maschinellen Übersetzung

- **Multi-reference word error rate (mWER) für vier Sprachmodelle mit unterschiedlicher Kontextlänge (Verbmobil Deutsch–Englisch):**

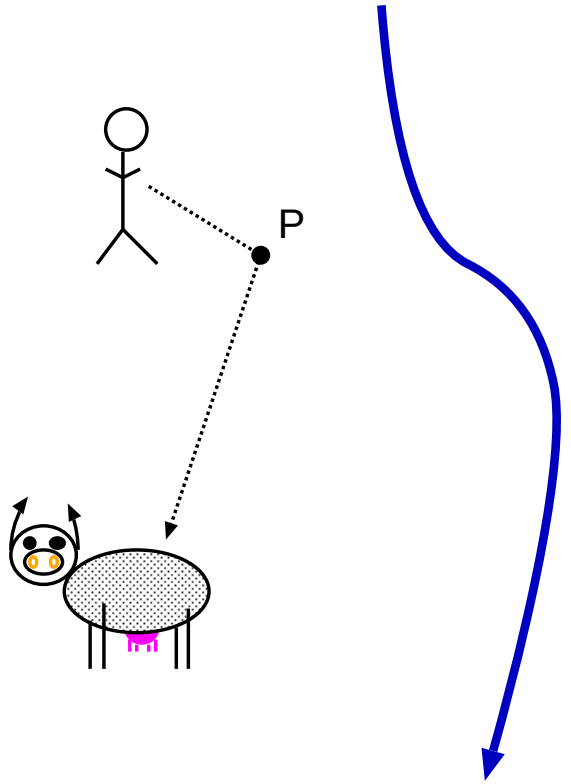
Sprachmodell	mWER[%]
0-gram	77.1
1-gram	38.8
2-gram	25.6
3-gram	24.7

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**$(P) = 0$

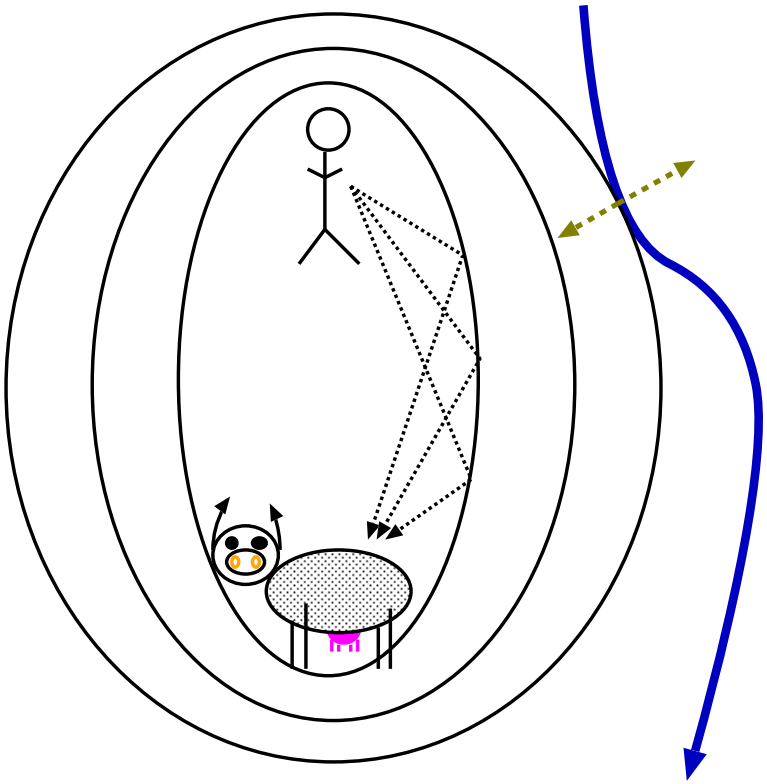


Lagrange Multiplikator

- **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 Logarithmus:**
  - **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:**

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

- **Partielle Ableitungen:**

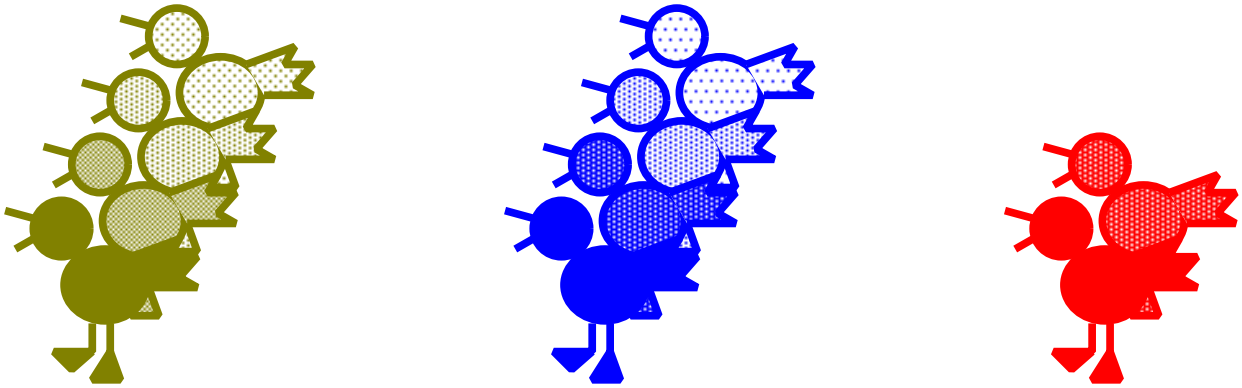
$$\frac{\partial \tilde{F}}{\partial p_\theta(w|h)} = \frac{N(h,w)}{p_\theta(w|h)} + \lambda_h = 0$$

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

- **Ergebnisse der Maximum Likelihood Schätzung:**

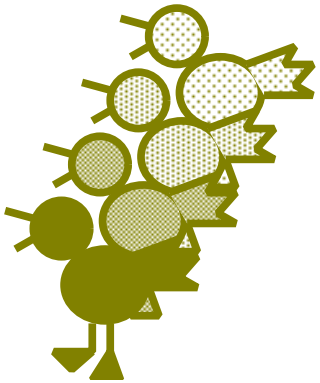
$$p(w|h) = \frac{N(h,w)}{\sum_{w'} N(h,w')} = \frac{N(h,w)}{N(h,\cdot)}$$

Spatz-Modell

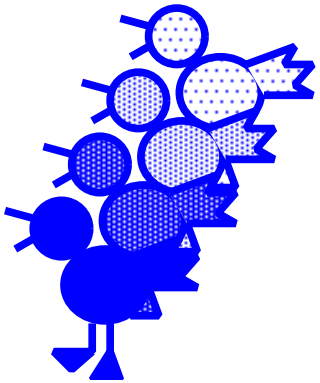


- **Wie wahrscheinlich ist ein Spatz der Farbe x?**

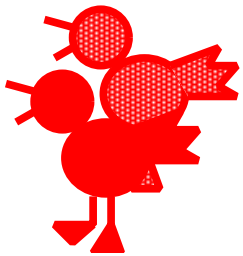
Spatz-Modell



40%



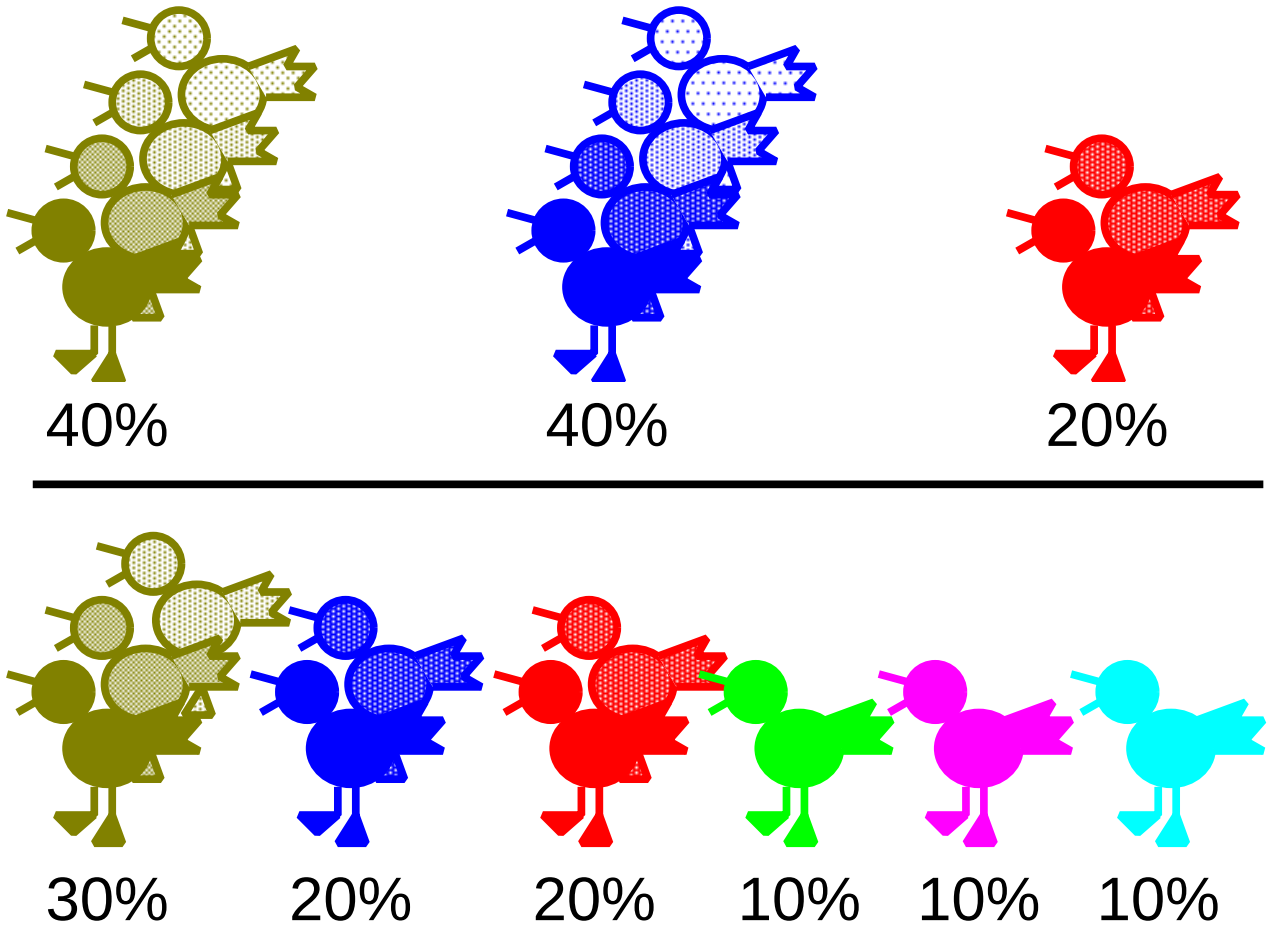
40%



20%

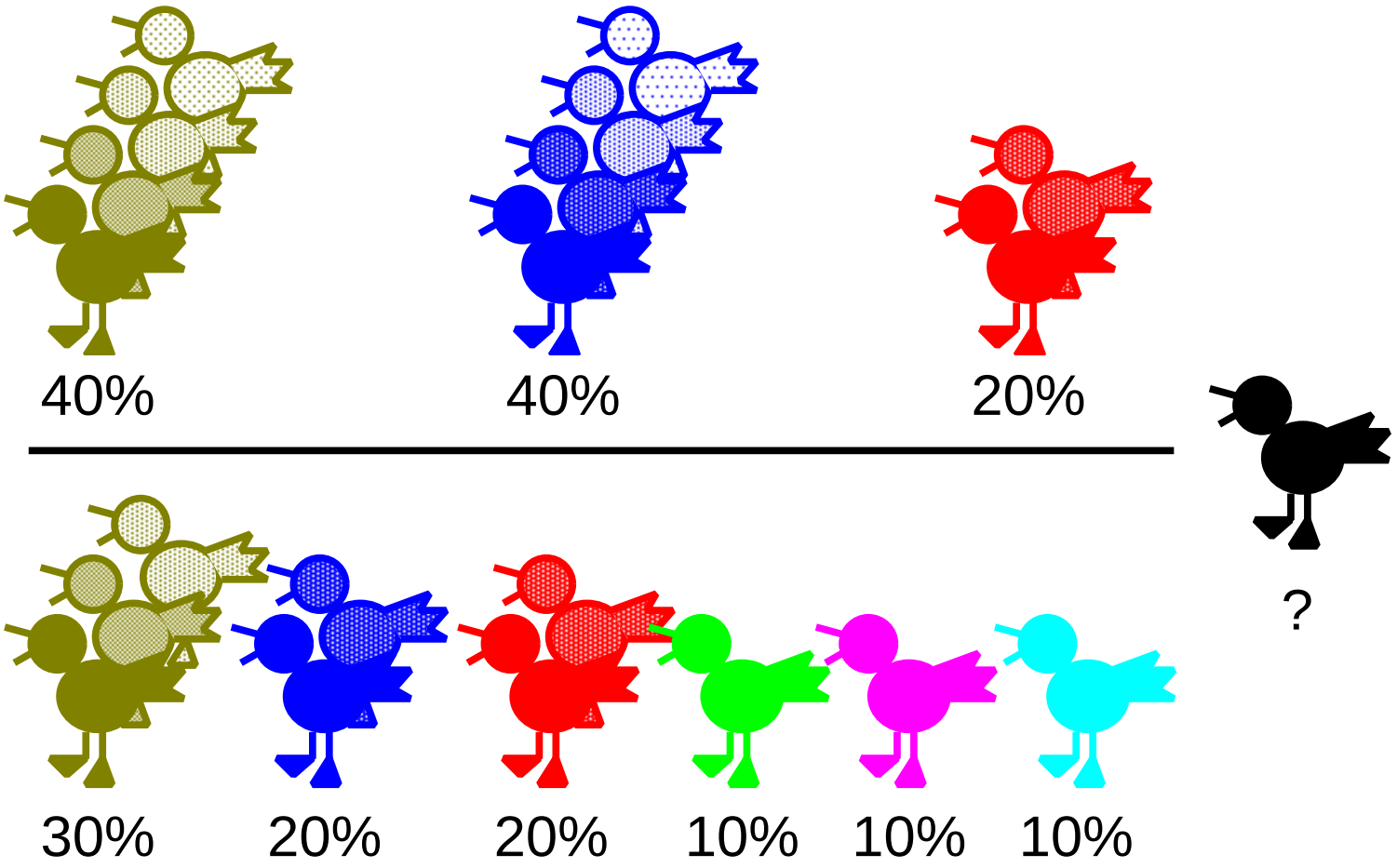
- **Konventioneller Ansatz: Relative Häufigkeit**

Spatz-Modell



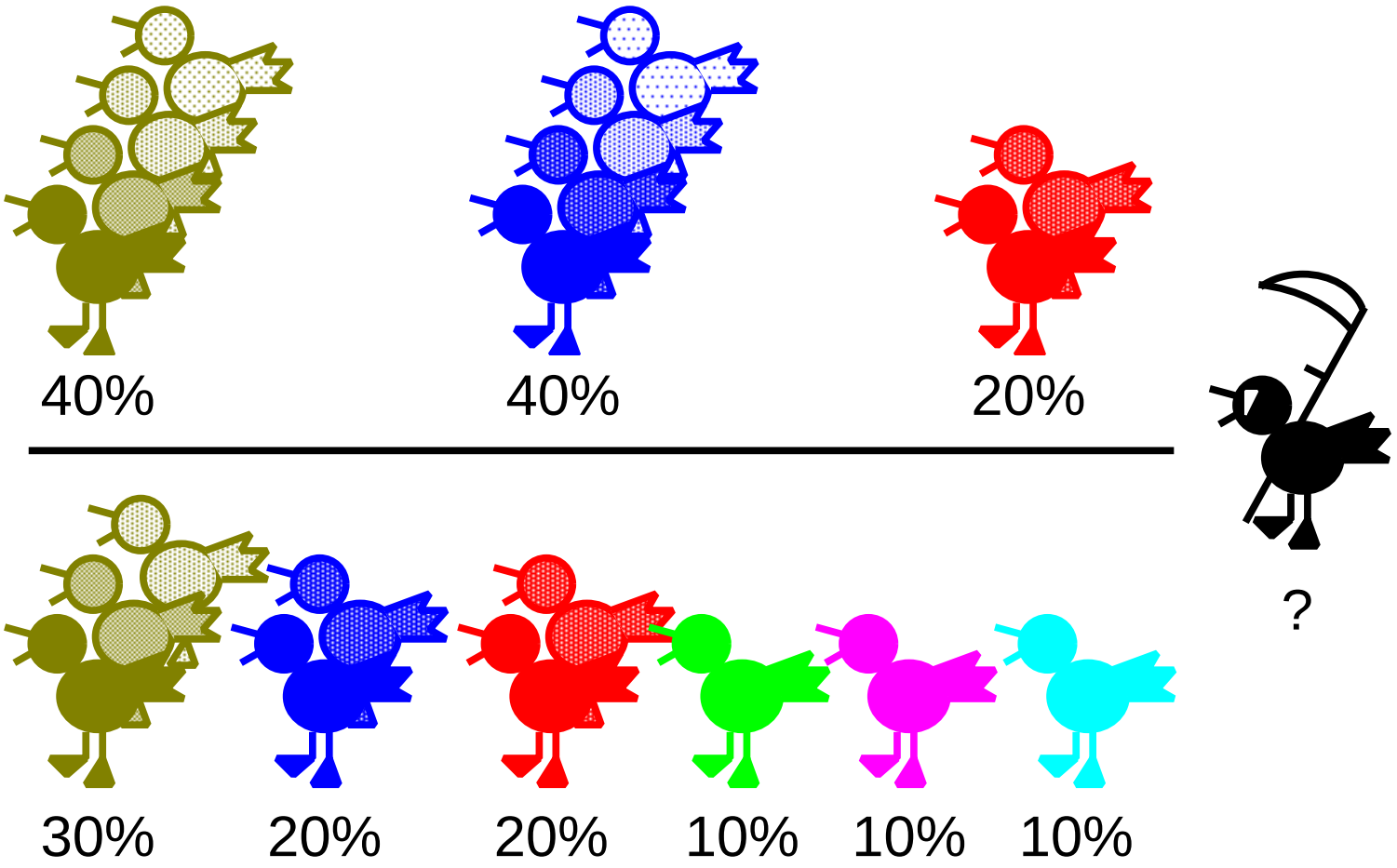
- **Welches von beiden Modellen ist stabiler?**

Spatz-Modell



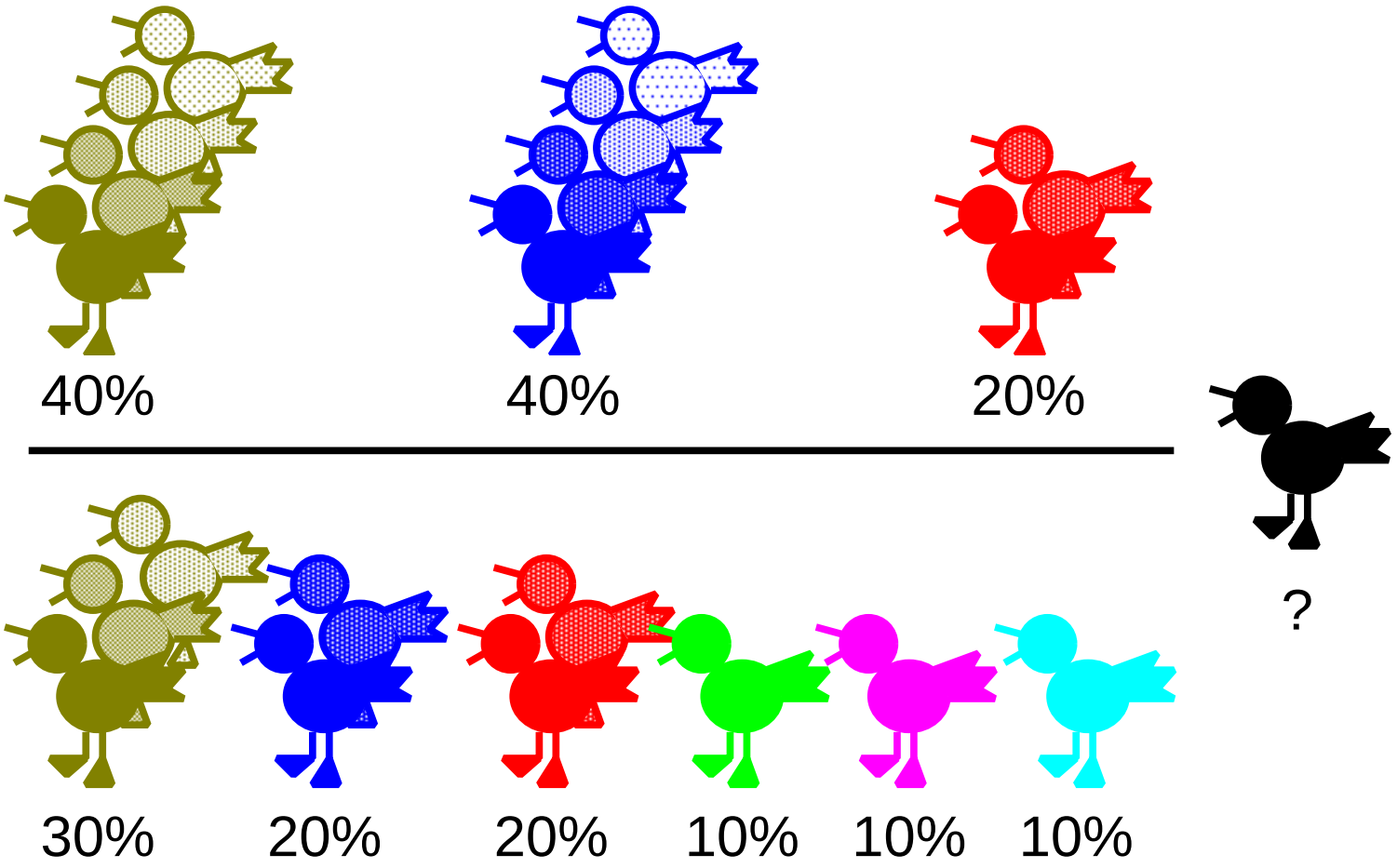
- **Was passiert mit ungesehenen Ereignissen? Wie wahrscheinlich sind sie?**

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:**
  - **Bigramm**
    Anzahl der möglichen Ereignisse: $|\text{Vokabular}|^2 = 4 \cdot 10^8$
    $\Rightarrow 10/400 = 2.5\%$ **können gesehen werden**
  - **Trigramm**
    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**
- **Problem:**
  - **Trainingskorpus enthält Trigramm 'Angela Merkel könnte', aber nicht 'Angela Merkel kann'**
  - **auch das zweite Trigramm sollte eine Wahrscheinlichkeit** > 0 **zugewiesen bekommen**

Linear Discounting

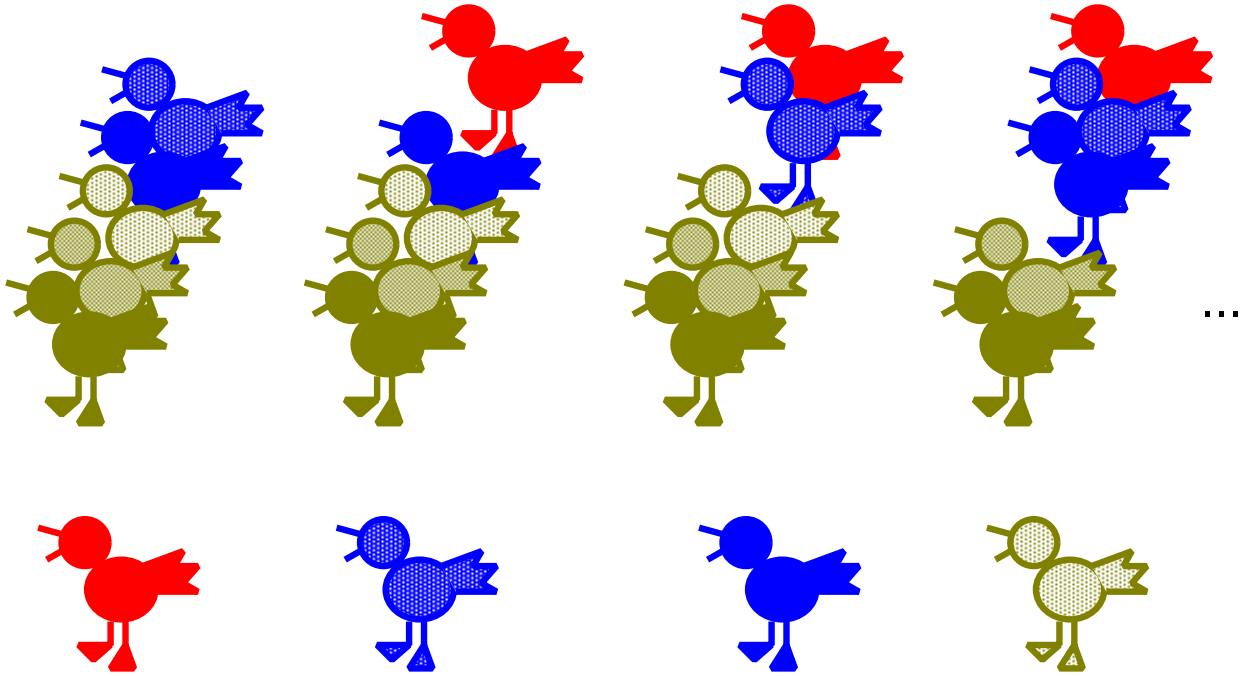
- **Discounting: Verschiebe einen Teil der Wahrscheinlichkeitsmasse zu ungesehenen Ereignissen**
- **Linear: Proportional für jeden Count $N(h, w)$**

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

- **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: $\mu = \frac{N_1(\cdot, \cdot)}{N(\cdot, \cdot)}$
- $\beta(w|\overline{h})$:
  - Sprachmodell mit kleinerer History $\overline{h}$

2 Übung 6

- **Trainieren eines Bigramm-Sprachmodells für Englisch mit Hilfe des SRI LM Toolkits**
- **Reranking der n-best-Listen mit Sprachmodell**

3 SRI LM Toolkit

- **SRI Language Modeling 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

</s>	Hunger	habe	Ich	<s>	Vocab_None

- 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)

Fragen?

Wenn ihr jetzt immer noch nicht wisst, wo unser Büro ist, haben wir was falsch gemacht :-)