Statistische Maschinelle Übersetzung

Teil II - Word Alignment

Thomas Schoenemann
Heinrich-Heine-Universität Düsseldorf

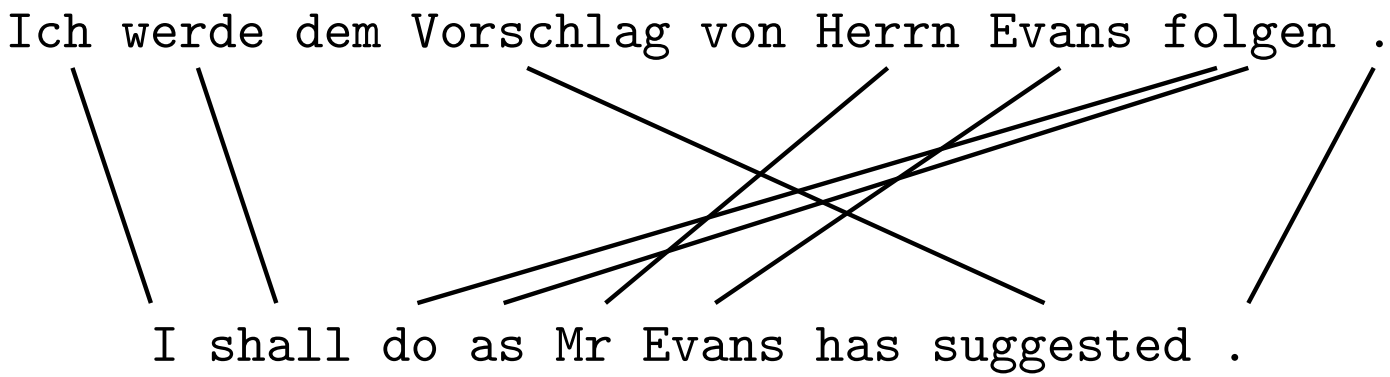
Sommersemester 2012

Zielsetzung

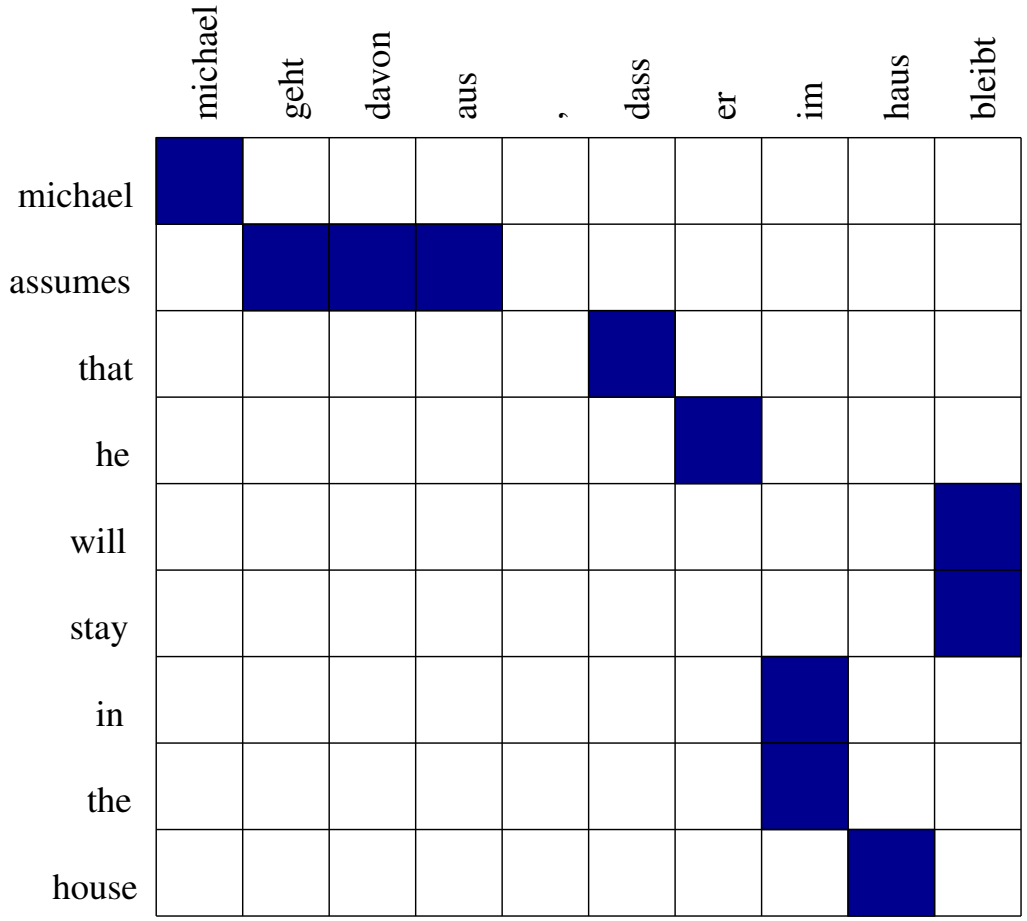
Gegeben: Zwei Sätze f_1^J, e_1^I

Aufgabe: Ermittele eine Menge von Links von Wörtern, die einander entsprechen.

Notation: Alignment $\mathbf{a} \subseteq \{1, \ldots, J\} \times \{1, \ldots, I\}$



Word Alignment: Alternative Darstellung



- - mehr Platzbedarf ++ besser bei stark nicht-monotonen Alignments

Word Alignment: Anwendungen

- Maschinelle Übersetzung
  – Alignments dienen als Initialisierung für komplexere Übersetzungsmodelle
- Monolinguales Data Mining (durch Berücksichtigung einer Fremdsprache), z.B.
  – Extraktion von Mehrwortausdrücken.
  – Finden von Wörtern mit mehreren Bedeutungen (und der entsprechenden Bedeutungen)

Word Alignment: Probabilistischer Ansatz

Gleiches Vorgehen wie bei der Gesamtstrategie für probabilistische Übersetzung:

- Entwerfe ein Modell, das (in Abhängigkeit von Parametern) jedem Alignment eine Wahrscheinlichkeit zuordnet.
- Ermittele geeignete Parameter aus Trainingsdaten.
- Für ein gegebenes Satzpaar wird nun das wahrscheinlichste Alignment ermittelt.
  – auf späteren Folien: auch andere Strategien (Minimum Bayes Risk)

Anmerkung: Die betrachteten Modelle für Word Alignment sind vollwertige Übersetzungsmodelle, werden dafür aber nicht mehr eingesetzt.

Einschub: nicht-probabilistischer Ansatz

Ein einfacher Ansatz:

- Training auf einer Folge von Satzpaaren $\mathbf{f}_s, \mathbf{e}_s, s = 1, \ldots, S$.
  Berechne die Zahl $N(\tilde{f}, \tilde{e})$, wie oft ein Wort $\tilde{f}$ in einem Satz $\mathbf{f}_s$ gleichzeitig mit dem Wort $\tilde{e}$ im Satz $\mathbf{e}_s$ auftritt.

$$N(\tilde{f}, \tilde{e}) = \sum_s \sum_{i=1}^{I_s} \sum_{j=1}^{J_s} \Big[\delta(f_j^s, \tilde{f})\, \delta(e_i^s, \tilde{e}) \Big]$$

  Kronecker-δ: $\delta(x, y) = 1$ falls $x = y$, 0 sonst.

- Bei der Berechnung eines Alignment für gegebene f_1^J, e_1^I, aligniere z.B. Wörter f_j und e_i wenn $N(f_j, e_i) \geq K$ für ein festes K.

In der Praxis: Modellbasierte Ansätze funktionieren besser
– auch Statistik auf den Alignments
– Das Grundprinzip der *Co-occurrence* (gleichzeitiges Auftreten) ist jedoch auch für diese Modelle zentral.

Unüberwachtes Training

Wir betrachten hier nur den Fall des *unüberwachten* Lernens (engl. unsupervised learning):

- keine Alignments gegeben, lediglich bilinguale Satzpaare.
- Basismodell:

$$p(f_1^J|e_1^I) = \sum_{\mathbf{a}} p(f_1^J, \mathbf{a} \,|\, e_1^I)$$

  - Alignments sind versteckte Variablen (engl. hidden)
  - bedingte Wahrscheinlichkeiten: e_1^I ist gegeben.
  - Es gibt i.A. $2^{I \cdot J}$ Alignments. Exakte Berechnung der Summe erfordert eine spezielle Struktur des Modells.

- auf späteren Folien auch $p(f_1^J, e_1^I)$

Anmerkung: In der Forschung wird auch überwachtes Lernen verfolgt. Jedoch sind nur selten ausreichend Alignments verfügbar.

Einzelwort-basierte Modelle (Brown et al. 1993, Vogel et al. 1996)

Gängigste Modelle für $p(f_1^J|e_1^I)$:
– Annahme: jedes Wort f_j entspricht höchstens einem Wort e_i.
– Notation: $a_j \in \{0, \ldots, I\}$, $a_j = 0$ heißt dass f_j nicht aligniert ist.
Technischer Kniff: füge $e_0 = \texttt{NULL}$ ein $\Rightarrow$ alle f-Wörter aligniert.
– $\mathbf{a} = a_1^J$.
– jetzt $(I+1)^J$ mögliche Alignments.

Unterteilung in
– sequenzbasierte Modelle: Faktorisierung entlang der f-Sequenz
– fertilitätsbasierte Modelle: Faktorisierung entlang der e-Sequenz
(engl. *fertility* = Fruchtbarkeit, s.u.)

Implementierungen: GIZA++, RegAligner (Berkeley Aligner: teilweise, aber mit Zusatztermen)

Sequenzbasierte Einzelwortmodelle

Erinnerung: $p(f_1^J|e_1^I) = \sum_{\mathbf{a}} p(f_1^J, a_1^J \,|\, e_1^I)$

Für sequenzbasierte Modelle:

$$p(f_1^J, a_1^J \,|\, e_1^I) = p(f_1^J \,|\, a_1^J, e_1^I) \cdot p(a_1^J|I)$$

Übersetzungswahrscheinlichkeit:

$$p(f_1^J \,|\, a_1^J, e_1^I) = \prod_{j=1}^{J} p(f_j \,|\, e_{a_j})$$

$p(f|e_0)$ ist die Wk. dass f ohne Alignment auftritt.

Alignmentwahrscheinlichkeit:

- IBM-1 : $p(a_1^J|I) = \prod_{j=1}^{J} \frac{1}{I+1} = \frac{1}{(I+1)^J}$ (alle Alignments gleich wahrscheinlich)
- HMM : $p(a_1^J|I) = p(a_1|I) \prod_{j=2}^{J} p(a_j \,|\, a_{j-1}, I)$

Hidden Markov Modelle: Markov Ketten

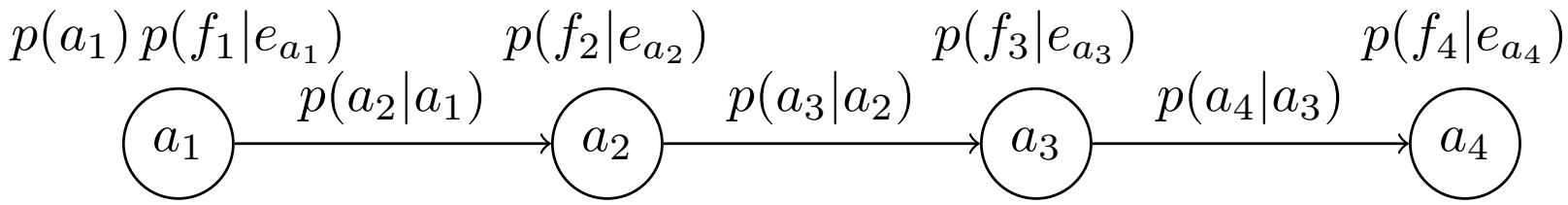
Hidden heißt, dass die Werte der Alignmentvariablen (*Zustände*) nicht aus den Eingabedaten ablesbar sind.

Gesamtmodell für das HMM:

$$p(f_1^J, a_1^J | e_1^I) = \left[\prod_{j=1}^J p(f_j|e_{a_j})\right] \cdot p(a_1|I) \prod_{j=2}^J p(a_j \,|\, a_{j-1}, I)$$

In dieser Form:

Das HMM für word alignment ist eine *Markov Kette* (1. Ordnung) :



Jedoch: Sonderbehandlung von `NULL`

HMM: Behandlung unalignierter Wörter

Sonderbehandlung für die Alignierung zu `NULL`:

– unalignierte Wörter treten relativ zufällig auf
Daher: $p(a_j = 0 \,|\, a_{j-1}, I) = p_0$ nicht abhängig von a_{j-1}, I

– falls $a_{j-1} = 0$, nehme nicht $p(a_j|a_{j-1}, I)$, sondern konditioniere auf die letzte *alignierte Position*:
$p(a_j|a_{j'}, I),\ j' = \max\{j'' < j \,|\, a_{j''} > 0\}$
wobei $\sum_{i=1}^{I} p(i|a_{j'}, I) = 1 - p_0$

– üblich: falls alle vorherigen Wörter unaligniert, lege eine Position fest (wird im Training mitgeschätzt)

Hidden Markov Modell: Integration von NULL

Ziel: Markov Kette 1. Ordnung *trotz* Sonderbehandlung von `NULL`

Dazu: Erweiterter Zustandsraum:
$a_j \in \{0,1\} \times \{1,\ldots,I\}$,
(Literatur: häufig kodiert als $a_j \in \{1,\ldots 2I\}, (b,i) \mapsto i + bI$)

erste Komponente: 0 = aligniert, 1 = nicht aligniert
zweite Komponente: Position in e, zu der das letzte vorhergehende *alignierte* f-Wort aligniert.

Jetzt: $p(a_1^J|I) = p(a_1|I) \prod_{j=2}^{J} p(a_j \,|\, a_{j-1}, I)$ drückt auch die Sonderbehandlung aus, denn:

$$p((0,i)|(\cdot,i')) = p_1 \cdot p(i|i',I)$$

$$p((1,i)|(\cdot,i')) = p_0 \cdot \delta(i,i')$$

p_0 und p_1 sind Parameter des Modells, $p_0 + p_1 = 1$ (werden im Training mitgeschätzt).

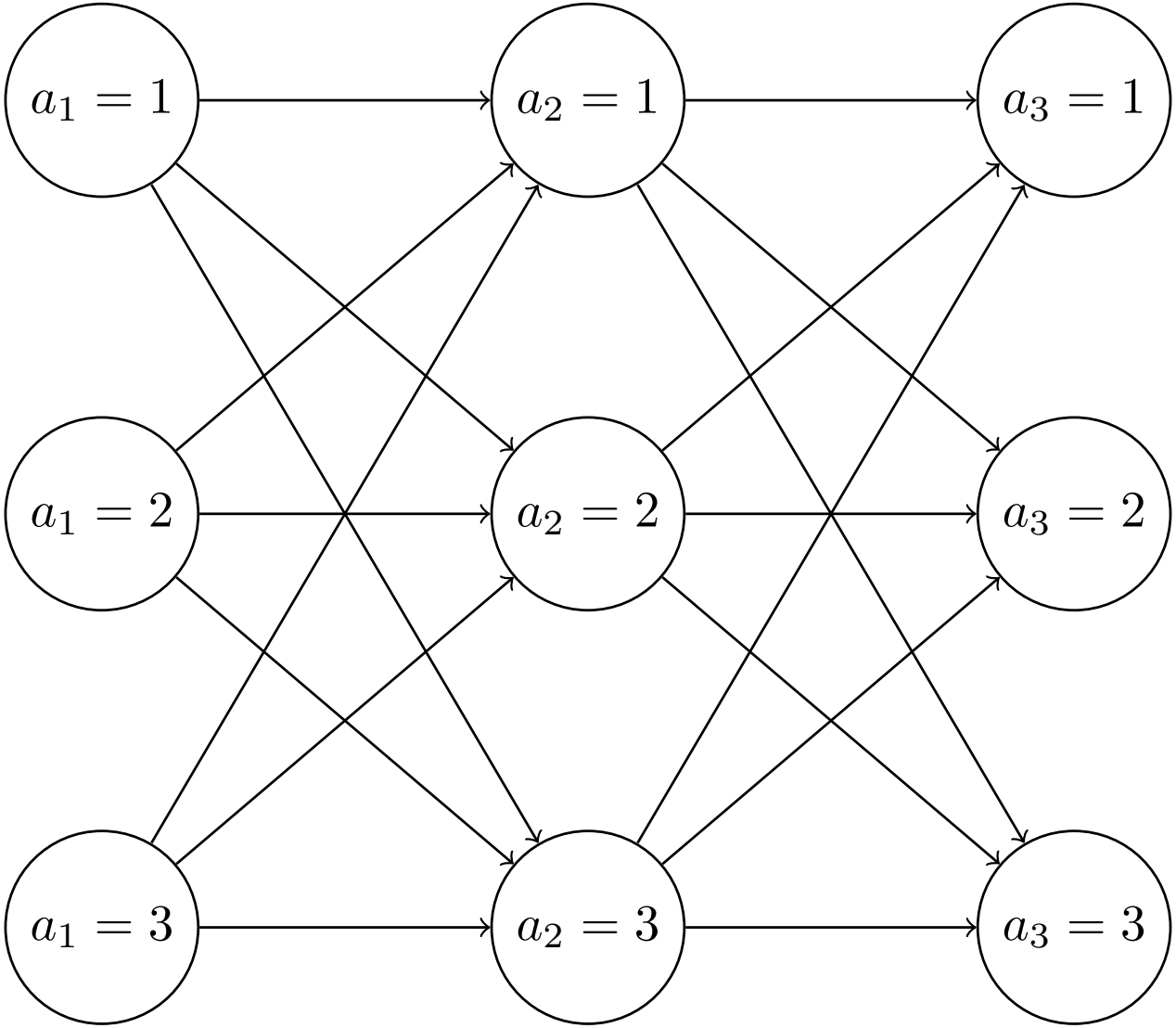
Hidden Markov Modell: Integration von NULL(2)

Bei Alignmentvariablen $a_j \in \{0,1\} \times \{1,\ldots,I\}$:

aligniertes Wort e_{a_j} ergibt sich wie folgt:

- Falls $a_j = (0,i)$: $\Rightarrow e_{a_j} = e_i$
- Sonst $(a_j = (1,i))$: $\Rightarrow e_{a_j} = e_0$

Hidden Markov Modelle: Zustandsgraph (ohne NULL)



Hidden Markov Modelle: Parametrisierung

Erinnerung: Gesamtmodell für das HMM:

$$p(f_1^J, a_1^J \,|\, e_1^I) = \left[\prod_{j=1}^{J} p(f_j \,|\, e_{a_j})\right] \cdot p(a_1|I) \prod_{j=2}^{J} p(a_j \,|\, a_{j-1}, I)$$

Einfachste Umsetzung für das Alignmentmodell:

– Parameter $p(i|i', I) \geq 0$ für alle Kombinationen von i, i', I.

– führt im Training zu *Overfitting*:

  – starke Anpassung an die Trainingsdaten.

  – schlechte Generalisierung auf ungesehene Daten.

Besser: differenzbasiert

– $p(i|i', I) = \frac{p(i-i')}{\sum_{i''=1}^{I} p(i''-i')}$

– häufig: Gruppierung großer Distanzen ($|i'' - i'| > 5$).

Erwartungswerte

Aufgabe: für gegebene f_1^J, e_1^I, und bekannte i, j, berechne

$$p(a_j = i|f_1^J, e_1^I) = \frac{1}{p(f_1^J|e_1^I)} \sum_{a_1^J : a_j = i} p(f_1^J, a_1^J|e_1^I)$$

Stichwort: marginale Wahrscheinlichkeit

Wichtig für: Training der Modelle (*expectation maximization*, s.u.)

Mittels $p(a_j = i|f_1^J, e_1^I)$ weiterhin berechenbar:

- Erwartungswert, dass für das gegeb. Paar f_1^J, e_1^I ein Wort $\tilde{f}$ zu einem Wort $\tilde{e}$ aligniert (Beachte: $\tilde{f}$ kann mehrmals (oder gar nicht) in f_1^J vorkommen. Analog für $\tilde{e}$):

$$E_{f_1^J, e_1^I}(\tilde{f} \leftrightarrow \tilde{e}) = \sum_{i,j} \Big[\delta(e_i, \tilde{e})\, \delta(f_j, \tilde{f})\, p(a_j = i|f_1^J, e_1^I) \Big]$$

Erwartungswerte für IBM-1

Zunächst: Berechnung von $p(f_1^J|e_1^I) = \sum_{a_1^J} p(f_1^J, a_1^J|e_1^I)$ für IBM-1:

$$\begin{aligned}
p(f_1^J|e_1^I) &= \frac{1}{(I+1)^J} \sum_{a_1=0}^{I} \sum_{a_2=0}^{I} \cdots \sum_{a_J=0}^{I} p(f_1|e_{a_1})\, p(f_2|e_{a_2}) \cdots p(f_J|e_{a_J}) \\
&= \frac{1}{(I+1)^J} \sum_{a_1=0}^{I} p(f_1|e_{a_1}) \sum_{a_2=0}^{I} \cdots \sum_{a_J=0}^{I} p(f_2|e_{a_2}) \cdots p(f_J|e_{a_J}) \\
&\cdots \\
&= \frac{1}{(I+1)^J} \prod_{j=1}^{J} \left(\sum_{a_j=0}^{I} p(f_j|e_{a_j}) \right)
\end{aligned}$$

Somit marginale Wk.:

$$p(a_j = i|f_1^J, e_1^I) = \frac{c \cdot p(f_j|e_i)}{\sum_{i'} c \cdot p(f_j|e_{i'})} = \frac{p(f_j|e_i)}{\sum_{i'} p(f_j|e_{i'})}$$

Erwartungswerte für HMM

Zur Vereinfachung: $a_0 = -1, \quad \Rightarrow p(a_1) = p(a_1|a_0)$

Hier: i steht tatsächlich für (f, i'), $f \in \{0, 1\}$ (z.B. $i = i' + fI$)

$$p(f_1^J, a_j = i|e_1^I) = \sum_{a_1^J : a_j = i} p(f_1^J, a_1^J|e_1^I)$$

$$= \left[\sum_{a_{j-1}} \ldots \sum_{a_1} p(i|a_{j-1}) \prod_{j'=1}^{j-1} \left[p(f_{j'}|e_{a_{j'}})\, p(a_{j'}|a_{j'-1})\right]\right]$$

$$\cdot\, p(f_j|e_i) \cdot \left[\sum_{a_{j+1}} \ldots \sum_{a_J} \left[\prod_{j'=j+1}^{J} p(f_{j'}|e_{a_{j'}})\right] p(a_{j+1}|i) \prod_{j'=j+2}^{J} p(a_{j'}|a_{j'-1})\right]$$

Problem zerfällt in zwei Teilprobleme.

Forward-Backward

Hilfsgrößen (für gegebene f_1^J, e_1^I):

- Forward-Wahrscheinlichkeit:

$$q_F^j(i) = p(f_j|e_i) \sum_{a_{j-1}} \ldots \sum_{a_1} p(i|a_{j-1}) \prod_{j'=1}^{j-1} \Big[p(f_{j'}|e_{a_{j'}})\, p(a_{j'}|a_{j'-1}) \Big]$$

- Backward-Wahrscheinlichkeit:

$$q_B^j(i) = p(f_j|e_i) \sum_{a_{j+1}} \ldots \sum_{a_J} \left[\prod_{j'=j+1}^{J} p(f_{j'}|e_{a_{j'}}) \right] p(a_{j+1}|i) \prod_{j'=j+2}^{J} p(a_{j'}|a_{j'-1})$$

Dann (falls $p(f_j|e_i) > 0$): $p(a_j = i|f_1^J, e_1^I) = \frac{q_F^j(i)\ q_B^j(i)}{p(f_1^J|e_1^I) p(f_j|e_i)}$

wobei $p(f_1^J|e_1^I) = \sum_i q_F^J(i)$

Berechnung von Forward

Erinnerung:

$$q_F^j(i) = p(f_j|e_i) \sum_{a_{j-1}} \ldots \sum_{a_1} p(i|a_{j-1}) \prod_{j'=1}^{j-1} \Big[p(f_{j'}|e_{a_{j'}})\, p(a_{j'}|a_{j'-1}) \Big]$$

Iterative Berechnung für Forward:

- $q_F^1(i) = p(f_1|e_i) \cdot p(a_1 = i)$
- für $j > 1$:
  $q_F^j(i) = p(f_j|e_i) \sum_{a_{j-1}} \Big[p(i|a_{j-1})\, q_F^{j-1}(a_{j-1}) \Big]$

Iterative Berechnung von $j = 1$ schrittweise aufwärts bis $j = J$.

Stichwort: dynamische Programmierung (jedoch hier kein Optimierungsproblem)

Berechnung von Backward

Für Backward gilt analog:

Erinnerung:

$$q_B^j(i) = p(f_j|e_i) \sum_{a_{j+1}} \ldots \sum_{a_J} \left[\prod_{j'=j+1}^{J} p(f_{j'}|e_{a_{j'}}) \right] p(a_{j+1}|i) \prod_{j'=j+2}^{J} p(a_{j'}|a_{j'-1})$$

Iterative Berechnung für Backward:

- $q_B^J(i) = p(f_J|e_i)$
- für $j < J$:
  $q_B^j(i) = p(f_j|e_i) \sum_{a_{j+1}} \left[p(a_{j+1}|i)\, q_B^{j+1}(a_{j+1}) \right]$

Iterative Berechnung von $j = J$ schrittweise abwärts bis $j = 1$.

Training: Maximum Likelihood

Gegeben: Folge von Satzpaaren $\mathbf{f}_s, \mathbf{e}_s, s = 1, \ldots, S$ (*ohne* Alignments). $\mathbf{f}_s$ hat die Länge J_s, $\mathbf{e}_s$ die Länge I_s.

Gesucht: Geeignete Parameter für die Alignmentmodelle.

Fasse alle Modellparameter in einem Vektor $\boldsymbol{\theta}$ zusammen. (IBM-1: $\boldsymbol{\theta}$ enthält exakt alle $p(\tilde{f}|\tilde{e})$)

Abhängigkeit von $p(\cdot)$ von $\boldsymbol{\theta}$ wird durch $p_{\boldsymbol{\theta}}(\cdot)$ symbolisiert.

Maximum Likelihood Kriterium:

$$\max_{\boldsymbol{\theta}} \prod_{s=1}^{S} p_{\boldsymbol{\theta}}(\mathbf{f}_s|\mathbf{e}_s)$$

u.d. Nebenbedingung, dass $\boldsymbol{\theta}$ Wahrscheinlichkeitsverteilungen modelliert.

Negative Log-Likelihood

Äquivalent zu ML und einfacher handhabbar (gleiche Bedingungen an $\boldsymbol{\theta}$):

$$\min_{\boldsymbol{\theta}} \sum_{s=1}^{S} -\log(p_{\boldsymbol{\theta}}(\mathbf{f}_s|\mathbf{e}_s))$$

Beispiel: IBM-1 (bis auf Konstante)

$$\min_{\{p(f|e)\}\geq 0} \sum_{s=1}^{S}\sum_{j=1}^{J_s} -\log\Big(\sum_{i=1}^{I_s} p(f_j|e_i)\Big)$$

$$\text{s.t.} \sum_{\tilde{f}} p(\tilde{f}|\tilde{e}) = 1 \quad \forall e$$

Für das IBM-1 ist das ML-Training ein konvexes Problem.

(Ohne Nebenbedingungen: Gradientenabstieg führt zum Minimum. Auf Probleme mit Wahrscheinlichkeiten generalisierbar.)

Training: Vorbetrachungen (für nichtparametrische Modelle)

- Wenn Alignments gegeben wären
  ⇒ setze Parameter zu relativen Häufigkeiten
- Da aber keine Alignments gegeben:
  – Erwartungswerte ersetzen die Counts in den relativen Häufigkeiten
  – Aber: für die Berechnung der Erwartungswerte muss man die Parameter kennen
  – Daher: initialisiere die Parameter, dann iteriere

Expectation Maximization (EM): Intuitive Erklärung

Für das IBM-1: EM iteriert die Schritte:

1. Für alle $\tilde{f}, \tilde{e}$ berechne die Erwartungswerte (auf der Folge der Sätze $\mathbf{f}_s, \mathbf{e}_s$):
$$w_{\tilde{f},\tilde{e}} = \sum_s E_{\mathbf{f}_s,\mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e})$$
2. Normalisiere die Erwartungswerte:
$$p(\tilde{f}|\tilde{e}) := \frac{w_{\tilde{f},\tilde{e}}}{\sum_{\tilde{f}'} w_{\tilde{f}',\tilde{e}}}$$

Wenn – wie bei dem IBM-1 – das ML-Kriterium ein konvexes Minimierungsproblem ist, findet EM ein (globales) Minimum.

EM(intuitiv) für nichtparametrische HMMs (ohne Null)

d.h. separate Parameter $p(i|i', I)$ für alle i, i', I.

Basis:

- $E_{f_1^J, e_1^I}(i \to i') = \sum_{j=1}^{J-1} p(a_j = i, a_{j+1} = i' | f_1^J, e_1^I)$
- $p(a_j = i, a_{j+1} = i' | f_1^J, e_1^I) = q_F^j(i) \cdot p(i'|i, I) \cdot q_B^{j+1}(i')$

EM für nichtparametrisches HMM: Iteriere

1. Berechne $w_{\tilde{f},\tilde{e}}$ wie für IBM-1, zusätzlich

$$w_{i,i',I} = \sum_{s: I_s = I} E_{\mathbf{f}_s, \mathbf{e}_s}(i \to i')$$

2. Normalisiere die Erwartungswerte:

$$p(\tilde{f}|\tilde{e}) := \frac{w_{\tilde{f},\tilde{e}}}{\sum_{\tilde{f}'} w_{\tilde{f}',\tilde{e}}} \quad , \quad p(i|i', I) := \frac{w_{i,i',I}}{\sum_{i''} w_{i'',i',I}}$$

EM: Zwischenbetrachtung

Bisher:

– Berechnung und Normalisierung von Erwartungswerten

– funktioniert wenn

  – das Modell *nichtparametrisch* ist *und*

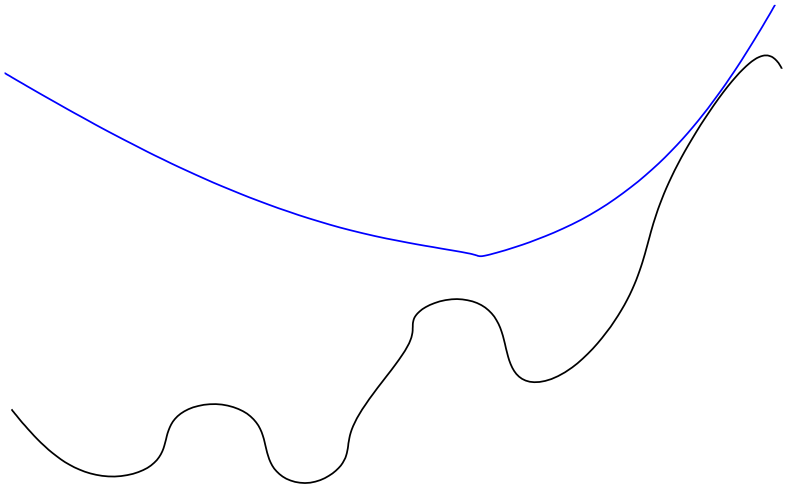
  – das Trainingskriterium Maximum Likelihood ist

Aber:

– EM kann viel mehr (parametrische Modelle, Regularisierungsterme)

– Bezug zum Maximum Likelihood Kriterium bisher diffus

EM: ein Majorize-Minimize Verfahren



Die ursprüngliche Neg LogLikelihood Funktion $g(\boldsymbol{\theta})$ (schwarz) ist schwer zu minimieren. Also:

1. wähle initialen Punkt $\boldsymbol{\theta}_0$
2. konstruiere eine Funktion $h(\boldsymbol{\theta})$ (blau), sodass $h(\boldsymbol{\theta}_0) = g(\boldsymbol{\theta}_0)$ und $h()$ obere Schranke für $g()$, d.h. $\forall \boldsymbol{\theta}' \quad h(\boldsymbol{\theta}') \geq g(\boldsymbol{\theta}')$. (`Majorize`)
3. wähle neuen Punkt $\boldsymbol{\theta}_0^{'}$ sodass $\boldsymbol{\theta}_0^{'}$ Minimum von $h()$ (ideal) oder zumindest $h(\boldsymbol{\theta}_0^{'}) < h(\boldsymbol{\theta}_0)$ (`Minimize`). Falls so ein Punkt existiert, setze $\boldsymbol{\theta}_0 := \boldsymbol{\theta}_0^{'}$ und gehe zu 2.

EM: Umformung der Negativen Log-Likelihood

Hier: p_θ steht für ein beliebiges Übersetzungsmodell (z.B. IBM-1, HMM).

Grundlage: $p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s) = \frac{p_{\boldsymbol{\theta}}(\mathbf{a}_s,\mathbf{f}_s|\mathbf{e}_s)}{p_{\boldsymbol{\theta}}(\mathbf{f}_s|\mathbf{e}_s)}$

Mit einer beliebigen (nicht modellbasierten)
Hilfs-Wahrscheinlichkeitsverteilung $q(\mathbf{a}_s)$ für jeden Satz s:

$$\begin{aligned}
\sum_s -\log p_{\boldsymbol{\theta}}(\mathbf{f}_s|\mathbf{e}_s) &= \sum_s \sum_{\mathbf{a}_s} -q(\mathbf{a}_s) \log p_{\boldsymbol{\theta}}(\mathbf{f}_s|\mathbf{e}_s) \\
&= \sum_s \Big[\sum_{\mathbf{a}_s} -q(\mathbf{a}_s) \log p_{\boldsymbol{\theta}}(\mathbf{a}_s,\mathbf{f}_s|\mathbf{e}_s) \Big] \\
&\quad + \Big[\sum_{\mathbf{a}_s} q(\mathbf{a}_s) \log p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s) \Big]
\end{aligned}$$

Wir behalten den ersten Term und ersetzen den zweiten durch eine obere Schranke (`Majorize`).

EM: Obere Schranke

Behauptung: $\Big[\sum_{\mathbf{a}_s} q(\mathbf{a}_s) \log p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s)\Big] \leq \Big[\sum_{\mathbf{a}_s} q(\mathbf{a}_s) \log q(\mathbf{a_s}))\Big]$

Beweis: es gilt $\log(x) \leq x-1 \quad \forall x > 0$, mit Gleichheit nur für $x = 1$. Somit

$$\begin{aligned}
& \Big[\sum_{\mathbf{a}_s} q(\mathbf{a}_s) \log p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s)\Big] - \Big[\sum_{\mathbf{a}_s} q(\mathbf{a}_s) \log q(\mathbf{a_s})\Big] \\
= & \sum_{\mathbf{a}_s} q(\mathbf{a}_s) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s)}{q(\mathbf{a_s})} \\
\leq & \sum_{\mathbf{a}_s} q(\mathbf{a}_s) \Big[\frac{p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s)}{q(\mathbf{a_s})} - 1\Big] \\
= & \Big[\sum_{\mathbf{a}_s} p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s)\Big] - \Big[\sum_{\mathbf{a}_s} q(\mathbf{a}_s)\Big] = 1 - 1 = 0
\end{aligned}$$

Gleichheit gilt falls $q(\mathbf{a}_s) = p_{\boldsymbol{\theta}}(\mathbf{a}_s|\mathbf{f}_s,\mathbf{e}_s)$.

EM: Majorize (und Minimize)

- Prinzipiell: obere Schranke für alle $q(\mathbf{a}_s)$.
- Aber: wir wollen $g(\boldsymbol{\theta}_0) = h(\boldsymbol{\theta}_0)$
  $\Rightarrow \quad q(\mathbf{a}_s) = p_{\boldsymbol{\theta}_0}(\mathbf{a}_s|\mathbf{f}_s, \mathbf{e}_s)$.

Die `Majorize`-Funktion $h(\boldsymbol{\theta})$ in einer Iteration von EM (mit aktuellem Punkt $\boldsymbol{\theta}_0$) ist somit:

$$h(\boldsymbol{\theta}) = \sum_s \Big[\sum_{\mathbf{a}_s} -p_{\boldsymbol{\theta}_0}(\mathbf{a}_s \,|\, \mathbf{f}_s, \mathbf{e}_s) \log p_{\boldsymbol{\theta}}(\mathbf{a}_s, \mathbf{f}_s|\mathbf{e}_s) \Big] + \texttt{const}$$

Um sie zu verwenden müssen zunächst die Erwartungswerte $p_{\boldsymbol{\theta}_0}(\mathbf{a}_s|\mathbf{f}_s, \mathbf{e}_s)$ berechnet werden (`E-step`).

Im `Minimize`-Schritt wird $h(\boldsymbol{\theta})$ über alle (zulässigen) $\boldsymbol{\theta}$ minimiert (`M-step`).

EM(formal) für IBM-1 : Majorize

Wiederholung: $h(\boldsymbol{\theta}) = \sum_s \Big[\sum_{\mathbf{a}_s} -p_{\boldsymbol{\theta}_0}(\mathbf{a}_s \,|\, \mathbf{f}_s, \mathbf{e}_s) \log p_{\boldsymbol{\theta}}(\mathbf{a}_s, \mathbf{f}_s | \mathbf{e}_s) \Big]$.

Summe über $(I+1)^J$ Alignments $\Rightarrow$ Struktur des Modells muss ausgenutzt werden.

Für IBM-1 ($\boldsymbol{\theta}$ enthält exakt alle $p(\tilde{f}|\tilde{e})$):

$$\begin{aligned}
h_{\texttt{IBM-1}}(\boldsymbol{\theta}) &= \sum_s \Bigg[\sum_{\mathbf{a}_s} -p_{\boldsymbol{\theta}_0}(\mathbf{a}_s \,|\, \mathbf{f}_s, \mathbf{e}_s) \sum_j \log(p(f_j^s | e_{a_j^s}^s)) \Bigg] \\
&= \sum_s \Bigg[\sum_j \sum_i -p(a_j^s = i | \mathbf{f}_s, \mathbf{e}_s) \log(p(f_j^s | e_i^s)) \Bigg] \\
&= \sum_{\tilde{f}} \sum_{\tilde{e}} -\Big[\sum_{s,i,j} \delta(f_j, \tilde{f})\, \delta(e_i, \tilde{e})\, p(a_j^s = i | \mathbf{f}_s, \mathbf{e}_s) \Big] \log p(\tilde{f}|\tilde{e}) \\
&= \sum_{\tilde{f}} \sum_{\tilde{e}} \sum_s -E_{\mathbf{f}_s, \mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e}) \log p(\tilde{f}|\tilde{e})
\end{aligned}$$

EM(formal) für IBM-1 : Minimize

$$\min_{\{p(\tilde{f}|\tilde{e})\}} \quad \sum_{\tilde{f}} \sum_{\tilde{e}} \sum_{s} -E_{\mathbf{f}_s,\mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e}) \log p(\tilde{f}|\tilde{e})$$

$$\text{sodass} \quad \sum_{\tilde{f}} p(\tilde{f}|\tilde{e}) = 1 \ \forall \tilde{e}\,, \quad p(\tilde{f}|\tilde{e}) \geq 0 \ \forall \tilde{f}, \tilde{e}$$

$\Downarrow$ (Lagrange Optimierung)

$$\max_{\boldsymbol{\lambda}} \min_{\{p(\tilde{f}|\tilde{e})\}} \sum_{f} \sum_{e} \sum_{s} -E_{\mathbf{f}_s,\mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e}) \log p\big(\tilde{f}|\tilde{e}\big) + \sum_{\tilde{e}} \lambda_{\tilde{e}} \Big(\sum_{f} p(\tilde{f}|\tilde{e}) - 1\Big)$$

$$\text{sodass} \quad p(\tilde{f}|\tilde{e}) \geq 0 \ \ \forall \tilde{f}, \tilde{e}$$

$\Downarrow$ (Ableitungen Nullsetzen)

$$\frac{\partial}{\partial p(\tilde{f}|\tilde{e})} \cdots = \frac{-\sum_s E_{\mathbf{f}_s,\mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e})}{p(\tilde{f}|\tilde{e})} + \lambda_{\tilde{e}} \quad \stackrel{!}{=} 0$$

$$\frac{\partial}{\partial \lambda_{\tilde{e}}} \cdots = \sum_{\tilde{f}} p(\tilde{f}|\tilde{e}) - 1 \quad \stackrel{!}{=} 0$$

EM(formal) für IBM-1 : Minimize (Lösung)

$$\frac{-\sum_s E_{\mathbf{f}_s,\mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e})}{p(\tilde{f}|\tilde{e})} + \lambda_{\tilde{e}} \quad \stackrel{!}{=} 0$$

$$\Leftrightarrow \quad \sum_s E_{\mathbf{f}_s,\mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e}) = \lambda_{\tilde{e}}\, p(\tilde{f}|\tilde{e})$$

$$\Leftrightarrow \quad p(\tilde{f}|\tilde{e}) = \frac{\sum_s E_{\mathbf{f}_s,\mathbf{e}_s}(\tilde{f} \leftrightarrow \tilde{e})}{\lambda_{\tilde{e}}}$$

Außerdem $\sum_{\tilde{f}} p(\tilde{f}|\tilde{e}) - 1 \Rightarrow$ wir erhalten die intuitive Erklärung

Anmerkungen:

- Nullsetzen der Ableitungen ist nur *notwendig*. (Hinreichend: müsste überprüfen, dass der gefundene Punkt ein Maximum für jedes $\lambda_{\tilde{e}}$ und ein Minimum für jedes $p(\tilde{f}|\tilde{e})$ ist).
- Der gefundene Punkt ist ein *innerer* Punkt: alle $p(\tilde{f}|\tilde{e})$ strikt positiv. Eigentlich müsste man auch Randpunkte ($p(\tilde{f}|\tilde{e}) = 0$ für ein Paar $\tilde{e}, \tilde{f}$) überprüfen.

EM-Training: Kette von Modellen

– EM benötigt *keine* Gradienten der ursprünglichen Funktion $g(\boldsymbol{\theta})$.

– EM ist frei von Schrittweiten (solange der Minimize-Schritt analytisch gelöst werden kann)

– EM hängt vom initialen Punkt $\boldsymbol{\theta}_0$ ab.

– EM findet generell ein *lokales* Minimum von $g(\boldsymbol{\theta})$,
wenn $g(\boldsymbol{\theta})$ konvex ein globales.

Für HMM (und später IBM-3 und IBM-4):

– nicht konvex, denn $\theta_1 \cdot \theta_2$ nicht konvex.

– gute Initialisierung:
erst EM-Training für IBM-1, dann HMM (dann IBM-3/4)

– generell: höherwertigere Alignments bei *wenigen* Iterationen (optimal: ca. 5)

Berechnung von Maximierenden Alignments

Problem: gegeben f_1^J, e_1^I, berechne das *Viterbi*-Alignment

$$\arg\max_{a_1^J} p(a_1^J|f_1^J, e_1^I) = \arg\max_{a_1^J} p(f_1^J, a_1^J|e_1^I)$$

Lösung:

– Für das IBM-1: $a_j^{\texttt{opt}} = \arg\max_{i\in\{0,\ldots,I\}} p(f_j|e_i)$

– Für das HMM:

Neue Hilfsgröße $\tilde{q}_F^j = \max_{a_1^j: a_j = i} \prod_{j'=1}^{j} p(f_j|e_{a_j}) p(a_j|a_{j-1})$

Iterativ berechenbar:

$$\tilde{q}_F^j(i) = p(f_j|e_i) \max_{a_{j-1}} p(i|a_{j-1})\, \tilde{q}_F^{j-1}(a_{j-1})$$

Extraktion des wahrscheinlichsten Alignments:

finde $\arg\max_i \tilde{q}_F^J(i)$, dann folge Backpointern

Alternativen zu Maximierenden Alignments

Maximierende Alignments werden berechnet, wenn das Training bereits abgeschlossen ist.

Ziel: jedem Satzpaar ein Alignment zuordnen

Alternativen zu Max. Alignments:

– Posterior Decoding: aligniere j und i, wenn $p(a_j = i \,|\, f_1^J, e_1^I) > \tau$, wobei τ Schwellwert ist.
Die entstehenden Alignments sind oft besser.
(sie müssen sich nicht an die Einzelwortannahme halten)

– Minimum Bayes Risk: aligniere j zu dem/einem i mit maximalem $p(a_j = i \,|\, f_1^J, e_1^I)$.

Schwächen von Sequenzbasierten Modellen

Bei IBM-1 und HMM:
– nur *alignierte* Wörter in e_1^I gehen in die Wahrscheinlichkeit ein.
– somit können Teile von e_1^I komplett ignoriert werden.

Bei fertilitätsbasierten Modellen (IBM-3 und IBM-4):
– alle Wörter in e_1^I explizit berücksichtigt.
– möglich da Faktorisierung entlang e
– jedoch: Modelle schwieriger handhabbar (Udupa und Maji, 2006, Brown et al. 1993)
  – Summe $p(f_1^J|e_1^I)$ über Alignments nicht effizient berechenbar
  – Erwartungswerte $p(a_j = i|e_1^I, f_1^J)$ nicht effizient berechenbar
  – Maximierende Alignments nicht effizient berechenbar
  – es gibt unmögliche "Alignments" mit $p(\cdot) > 0$ (s.u.)

Fertilitätsbasierte Modelle

Gegeben: Ein Satz e_1^I

Generiere Alignment und f-Satz nach folgendem Schema (engl. *generative story*)

1. Für $i = 1, \ldots, I$, bestimme die Anzahl Φ_i der f-Wörter, die zu e_i alignieren. Φ_i heißt *Fertilität* von Wort e_i (engl. fertility = Fruchtbarkeit). Wähle Φ_i mit Wahrscheinlichkeit $p(\Phi_i|e_i)$
2. Wähle nun die Fertilität Φ_0 von NULL mit Wk. $p(\Phi_0|\sum_{i=1}^{I}\Phi_i)$
3. Für jedes i (ab $i = 1$) und jedes $1 \leq k \leq \Phi_i$ wähle die alignierte Position d_{ik} im f-Satz. (*invertiertes Alignmentmodell*). Wähle mit Wahrscheinlichkeit
   $p(d_{ik}|i, \{d_{i'k'} \mid i' < i \text{ oder } (i' = i, k' < k)\}, J = \sum_{i=0}^{I}\Phi_i)$.
   Die verbleibenden Positionen alignieren zu NULL.
4. Für jedes i (einschl. $i = 0$) und $1 \leq k \leq \Phi_i$ wähle nun die Identität des alignierten f-Wortes $f_{d_{ik}}$ mit Wahrscheinlichkeit $p(f_{d_{ik}}|e_i)$.

Fertilitätsbasierte Modelle: Constraints

Nach Schritt 2 ist die Länge J des f-Satzes bekannt:

$$J = \sum_{i=0}^{I} \Phi_i$$

- Bei der Wahl von d_{ik} müssen alle vorherigen Assignments berücksichtigt werden (da jedes f_j nur einmal aligniert, und damit keine Lücken bleiben)
- Jedoch: nur im Modell IBM-5 umgesetzt (selten benutzt).
- Für IBM-3 und IBM-4: Constraint ignoriert
  $\Rightarrow$ unmögliche Alignments erhalten positive Wahrscheinlichkeit, das Modell summiert zu weniger als 1 (über die tatsächlich möglichen Alignments)
  (Fachausdruck (engl.): *deficient*)

Wahrscheinlichkeiten für Fertilitäten

Für die Wahrscheinlichkeit $p(\Phi_0|J' = \sum_{i=1}^{I} \Phi_i)$:

Jedes der J' alignierten f-Wörter generiert mit Wk. p_0 ein unaligniertes Wort. Somit:

$$p(\Phi_0|J') = \left(\begin{array}{c} J' \\ \Phi_0 \end{array} \right) p_0^{\Phi_0} \, (1 - p_0)^{J' - \Phi_0}$$

Och & Ney 2003: besser $p'(\Phi_0|J') = \frac{1}{J^{\Phi_0}} p(\Phi_0|J')$
Grund: sonst NULL-Modell non-deficient, führt im Training zu hohen p_0.

Die Wahrscheinlichkeiten $p(\Phi_i|e_i)$ sind schlicht nicht-parametrische Modelle.

Das Modell IBM-3

Modell 0. Ordnung:
$p(d_{ik}|i, \{d_{i'k'} \,|\, i' < i \text{ oder } i' = i, k' < k\}) = p(d_{ik}|i)$

$\Rightarrow$ Wk. für ein invertiertes Alignment:

$$p(f_1^J, \mathbf{d}|e_1^J) = p\Big(\Phi_0 \Big| \sum_{i=1}^{I} \Phi_i\Big) \cdot \prod_{k=1}^{\Phi_0} p(f_{d_{0,k}}|e_0) \cdot \prod_{i=1}^{I} \Big(p(\Phi_i|e_i) \prod_{k=1}^{\Phi_i} \big[p(f_{d_{ik}}|e_i)\, p(d_{ik}|i, J)\big]\Big)$$

Permutation der d_{ik} für ein fixes i führt zum gleichen (nicht-invertierten) Alignment a_1^J. Es gibt $\Phi_i!$ solche Permutationen. Somit Wk. für ein a_1^J:

$$p(f_1^J, a_1^J|e_1^J) = p\Big(\Phi_0 \Big| \sum_{i=1}^{I} \Phi_i\Big) \prod_{i=1}^{I} \big[\Phi_i!\, p(\Phi_i|e_i)\big] \cdot \prod_{j=1}^{J} \big[p(j|a_j, J)\, p(f_j|e_{a_j})\big]$$

Das Modell IBM-4

Hauptänderung zum IBM-3: Modell erster Ordnung:
$p(d_{ik}|i, \{d_{i'k'} \mid i' = \texttt{prev}(i) \text{ oder } i' = i, k' < k\}, J)$
wobei $\texttt{prev}(i)$ das größte $i' < i$ mit $\Phi_{i'} > 0$ ist (also das nähste vorherige alignierte e-Wort).

Zusatzannahme: $d_{ik} < d_{ik'}$ wenn $k < k'$.

($\Rightarrow$ keine Permutationen mehr).

Grundmodell: zwei Fälle:

1. $k = 1$: $p_{=1}(d_{i1}|\{d_{\texttt{prev}(i)k'}\}, J)$,
   Reduktion der Menge $\{d_{\texttt{prev}(i)k'}\}$ auf eine einzelne Position $\odot_{\texttt{prev}(i)}$.
   Typisch: gerundeter Mittelwert. RegAligner (default): Minimum.
   $\Rightarrow p(d_{i1} - \odot_{\texttt{prev}(i)})$ (parametrisches Modell)
2. $k > 1$: $p_{>1}(d_{ik}|d_{i(k-1)}, J)$. Üblich: parametrisches Modell
   $\Rightarrow p_{>1}(d_{ik} - d_{i(k-1)})$

IBM-4 und Wortklassen

Die Wörter der f-Sprache werden in Klassen $\mathcal{A}(f)$ unterteilt, die der e-Sprache in Klassen $\mathcal{B}(e)$. (Denkbar: Unterteilung nach POS-Tags. In der Praxis: monoling. maschinelles Lernen, 50 Klassen.)

1. $k = 1$: $p(d_{i1} - \odot_{\texttt{prev}(i)} | \mathcal{A}(f_{d_{i1}}), \mathcal{B}(e_{\texttt{prev}(i)}))$
2. $k > 1$: $p_{>1}(d_{ik} - d_{i(k-1)} | \mathcal{A}(f_{d_{ik}}))$.

Gesamtmodell IBM-4 (jetzt eindeutiges Mapping $a_1^J \mapsto \{d_{ik}\}$):

$$\begin{aligned} p(f_1^J, a_1^J | e_1^J) \quad = \quad & p\Big(\Phi_0 \Big| \sum_{i=1}^{I} \Phi_i\Big) \prod_{i=1}^{I} p(\Phi_i | e_i) \prod_{j=1}^{J} p(f_j | e_{a_j}) \\ & \prod_{i \geq 0 : \Phi_i > 0} \Big(p(d_{i1} - \odot_{\texttt{prev}(i)} | \mathcal{A}(f_{d_{i1}}), \mathcal{B}(e_{\texttt{prev}(i)})) \\ & \cdot \prod_{k=2}^{\Phi_i} p_{>1}(d_{ik} - d_{i(k-1)} | \mathcal{A}(f_{d_{ik}})) \Big) \end{aligned}$$

Anmerkung: IBM-4 und Wortklassen

Anmerkung: um das IBM-4 mit Wortklassen herzuleiten muss der Generierungsprozess angepasst werden. Die Schritte 3. und 4. müssen ersetzt werden durch die folgenden Regeln:

3. Für $i = 1, \ldots, I$ und $k = 1, \ldots \Phi_i$

   a) Generiere das Quellwort f_{ik} mit Wahrscheinlichkeit $p(f_{ik}|e_i)$.

   b) Generiere die Position d_{ik} des Wortes f_{ik} mit Wahrscheinlichkeit $p(d_{ik}|d_{i(k-1)}, f_{ik}, e_{\texttt{prev}(i)})$

4. Die verbleibenden Positionen alignieren zu NULL. Generiere die Wörter f_{0k} für $k = 1, \ldots, \Phi_0$ mit Wahrscheinlichkeit $p(f_{0k}|NULL)$.

Training von Fertilitätsbasierten Modellen

Anwendung von *expectation maximization* auf IBM-3 und IBM-4:

- Problem: die Erwartungswerte $p(a_j = i)$ und $p(\Phi_i = k)$ lassen sich nicht effizient berechnen (Udupa & Maji 2006)
- Approximation: Nehme maximierendes Alignment $\arg\max_{a_1^J} p(a_1^J | f_1^J, e_1^I)$ und seine Nachbarn :
  - Moves: $\hat{a}_1^J = a_1^J[j \mapsto i]$, wobei $i \neq a_j$: $\hat{a}_j = i$, sonst $\hat{a}_{j'} = a_{j'}$
  - Swaps: $\hat{a}_1^J = a_1^J[j_1 \leftrightarrow j_2]$, wobei $a_{j_1} \neq a_{j_2}$: $\hat{a}_{j_1} = a_{j_2}$, $\hat{a}_{j_2} = a_{j_1}$, sonst $\hat{a}_{j'} = a_{j'}$
- Aber: Auch das maximierende Alignment lässt sich nicht effizient berechnen (Udupa & Maji 2006) , es sei denn P=NP
  $\Rightarrow$ weitere Approximation: Hillclimbing

Hillclimbing für Maximierende Alignments

- Startpunkt: maximierendes HMM-Alignment (GIZA++) oder Alignment aus der vorherigen EM-Iteration (RegAligner)
- Iteriere: besuche alle Nachbarn (Moves und Swaps) und gehe zu dem Nachbarn mit der größten Wahrscheinlichkeit, falls diese höher als die initiale ist. (Ansonsten beende die Iteration)
- Für Effizienz: inkrementelle Wk. - Berechnung
  - nur sehr wenige Terme der Wk. ändern sich bei einem Move oder Swap. $\Rightarrow$

$$\texttt{neue Wk.} = \texttt{alte Wk.} \cdot \frac{\texttt{neu erscheinende Terme}}{\texttt{verschwindende Terme}}$$

  - für IBM-3: leicht zu implementieren
  - für IBM-4: kompliziert, aber möglich

Alternativ: Greedy Algorithm

– gehe jeweils zum ersten Nachbarn mit höherer Wahrscheinlichkeit.

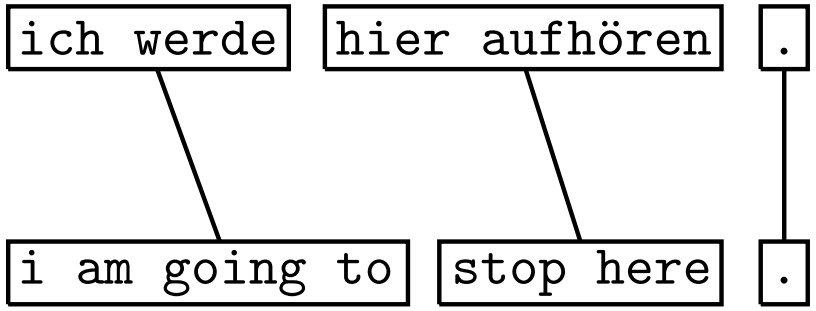
Schwächen von bedingten Wahrscheinlichkeiten für Word Alignment

- Seltene e-Wörter alignieren über Proportion zu f-Wörtern in den entsprechenden Sätzen (*garbage collection*).
- Grund: Sie haben die gleiche Wk.-Masse wie häufige Wörter, müssen aber nur einen Bruchteil ihrer eigentlichen Korrespondenzen erklären.
- Problem wird schlimmer wenn f-Wörter zu mehr als einem e-Wort alignieren dürfen.

Ausweg: Verbundverteilungen $p(\mathbf{f}, \mathbf{e})$ anstatt $p(\mathbf{f}|\mathbf{e})$
– bessere Modelle
– Aber: schwieriger zu handhaben (Erwartungswerte können meist nicht exakt berechnet werden).

Verbundverteilungsmodelle

Verbundverteilung für konsekutive Wortfolgen (Marcu and Wong 2002).



Generativer Prozess für e_1^I und f_1^J zusammen:

1. Bestimme eine Anzahl K von Phrasenpaaren (uniforme Verteilung)
2. Für jedes Phrasenpaar k, bestimme eine $\vec{e}$-Phrase $\vec{e}_k$ und eine $\vec{f}$-Phrase $\vec{f}_k$ mit Wk. $p(\vec{e}_k, \vec{f}_k)$.
3. Ordne die $\vec{e}_k$ zu einer Permutation π_e, ebenso π_f für die $\vec{f}_k$.
   – Modell MW-1: uniforme Verteilung (ignoriert)
   – Modell MW-2: Positionsmodell (in dieser Vorlesung nicht betrachtet)

Das Modell MW-1

Definition: $(K, \{\vec{e}_k, \vec{f}_k, | k = 1, \ldots, K\}, \pi_e, \pi_f) \in G(e_1^I, f_1^J)$ wenn der entsprechende Generierungsprozess e_1^I und f_1^J ergibt.

Wahrscheinlichkeit von e_1^I, f_1^J:

$$p(e_1^I, f_1^J) = \sum_{(K, \{\vec{e}_k, \vec{f}_k\}, \pi_e, \pi_f) \in G(e_1^I, f_1^J)} \prod_{k=1}^{K} p_{\texttt{phrase}}(\vec{e}_k, \vec{f}_k)$$

EM-Training:

– approximativer E-step, idealerweise über Nachbarschaft des maximierenden Alignments

– suboptimale Suche nach max. Alignment via Hillclimbing

Probleme:

– das Modell tendiert zu langen Phrasen, d.h. Maximum Likelihood setzt $p_{\texttt{phrase}}(\mathbf{e}_s, \mathbf{f}_s) > 0$, sonst alle $p_{\texttt{phrase}}(\vec{e}, \vec{f}) = 0$.

$\Rightarrow$ Abhilfe: geändertes Trainingskriterium (Regularisierungsterme)

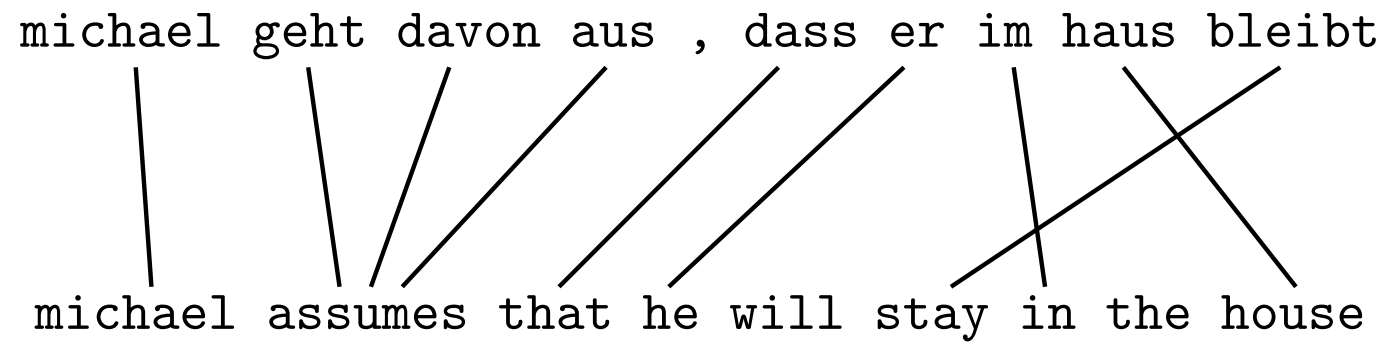
Symmetriesierung von Alignments

Einzelwortmodelle:
– jedes f-Wort kann nur einem e-Wort entsprechen
– wird realen Daten nicht gerecht.

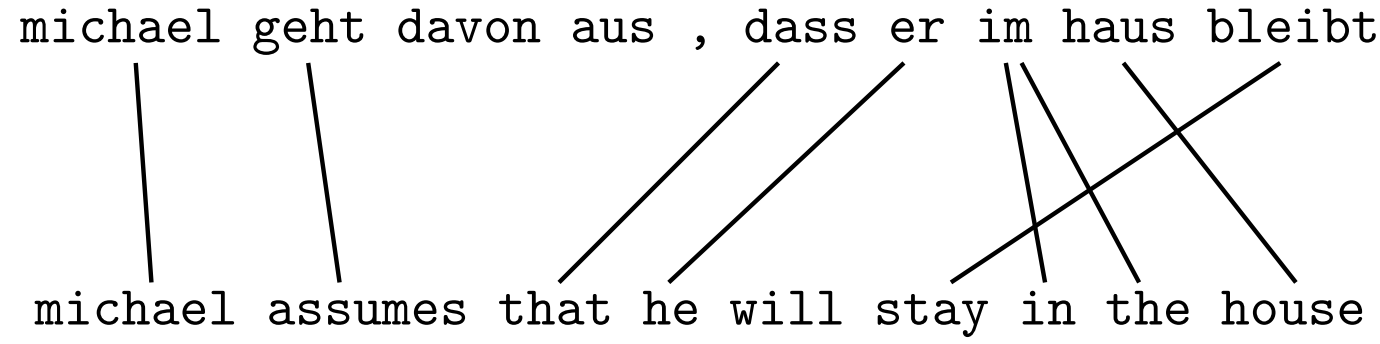
Deshalb:
– trainiere Modelle in beiden Richtungen ($p(\mathbf{f}|\mathbf{e})$ und $p(\mathbf{e}|\mathbf{f})$)
– kombiniere die Viterbi-Alignments
  – z.B. : Vereinigung
  – Standardmethode: **diag-grow-final-and**

Beispiel

$p(\mathbf{d}|\mathbf{e}) \rightarrow$



$p(\mathbf{e}|\mathbf{d}) \rightarrow$



diag-grow-final-and

Gegeben: zwei Alignments `e2f` und `f2e` für den gleichen Satz
Ausgabe: ein Alignment `a`.

1. **diag**: Starte mit dem Schnitt der beiden Alignments:
   `a` $=$ `e2f` $\cap$ `f2e`

2. **grow**: Iteriere:
   wenn $(j, i) \in$ `a`, besuche alle (j', i') mit $(j, i) - (j', i') \in$
   $\{(-1, 0), (0, -1), (1, 0), (0, 1), (-1, -1), (-1, 1), (1, -1), (1, 1)\}$.
   Wenn j' oder i' bisher ohne Alignment (in `a`) sind und $(j', i') \in$
   `e2f` $\cup$ `f2e`, füge (j', i') zu `a` hinzu.

3. **final-and** : wenn $(j', i') \in$ `e2f` $\cup$ `f2e` und j' oder i' bisher ohne Alignment (in `a`) sind, füge (j', i') zu `a` hinzu.

Wort- vs. Satzalignment

- Word Alignment setzt voraus, dass zusammengehörige Sätze bekannt sind. (tatsächlich werden oft mehrere Sätze zusammengefasst, da menschliche Übersetzer die Satzstruktur selten beibehalten.)
- In `Europarl v6` sind solche Segmentpaare gegeben (aber automatisch bestimmt worden).
- Normalerweise: nur zusammengehörige Texte bekannt
  $\Rightarrow$ zunächst Satzalignment notwendig

Satzalignment

- beide Texte werden anhand von Satzzeichen in Sätze unterteilt
- modellbasierte Ansätze um Zuordnung herzustellen
  - Erhaltung der Reihenfolge
  - Basiswahrscheinlichkeit, dass ein/zwei **f**-Sätze zu ein/zwei **e**-Sätzen zugeordnet werden
  - hilfreich: Eigennamen und bilinguale Lexika

Nachverarbeitung:

– entferne alle Paare wo eine Hälfte leer ist (kommt vor!)

– entferne alle Paare wo der **e**-Satz mehr als 9-mal länger ist als der **f**-Satz (oder umgekehrt).