Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Statistische Sprachmodelle

Tobias Scheffer
Paul Prasse
Michael Großhans

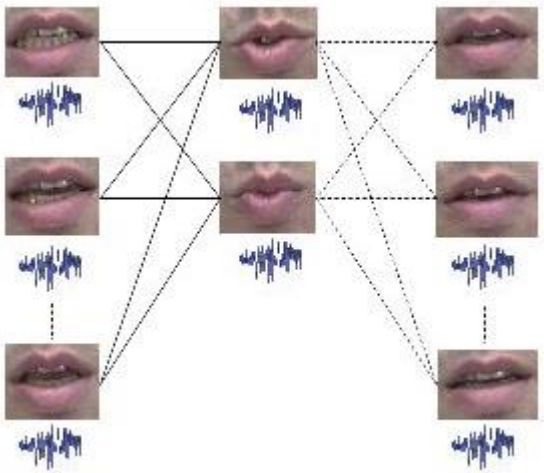
Statistische Sprachmodelle

- Welche Sätze sind Elemente einer Sprache.
  - Durch Grammatik einer Sprache beschrieben.
- Wie wahrscheinlich ist ein bestimmter Satz in einem Kontext:
  - P(„ich pflücke Beeren“) ist vielleicht 0.0001,
  - P(„ich pflücke Bären“) ist vielleicht 10^{-25}.
- Statistische Sprachmodelle:
  - N-Gramme,
  - Probabilistische kontextfreie Grammatiken.

Statistische Sprachmodelle
Wozu?

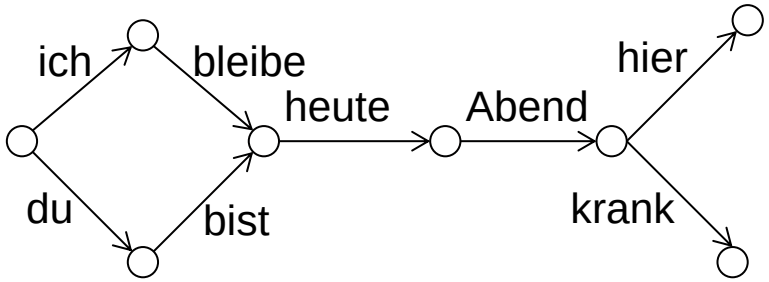
- Spracherkennung:
  - Akustisches Modell + Sprachmodell.

Phone



→ Wörter

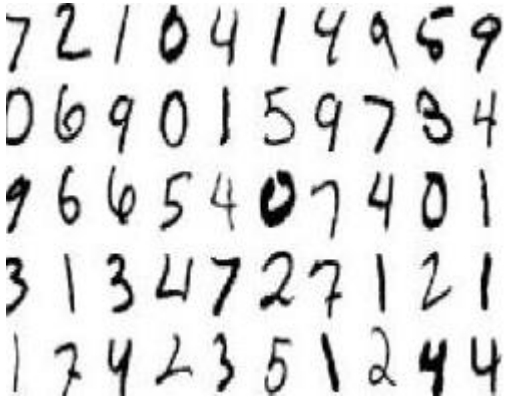
Sprachmodell



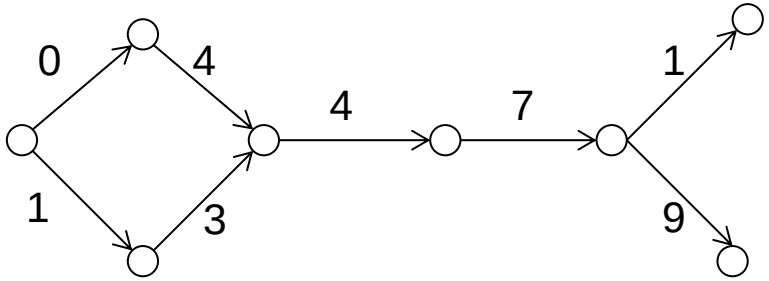
Statistische Sprachmodelle
Wozu?

- Handschrifterkennung:
  - Schreibmodell + Sprachmodell.

Schreibmodell



Sprachmodell

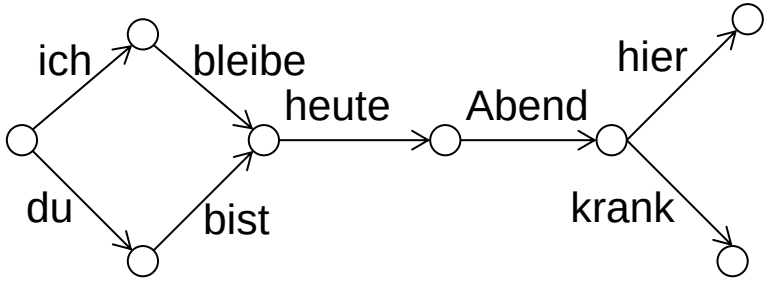


Statistische Sprachmodelle
Wozu?

- Rechtschreibkorrektur:
  - Fehler- / Vertippmodell + Sprachmodell.

Vertippmodell

Sprachmodell

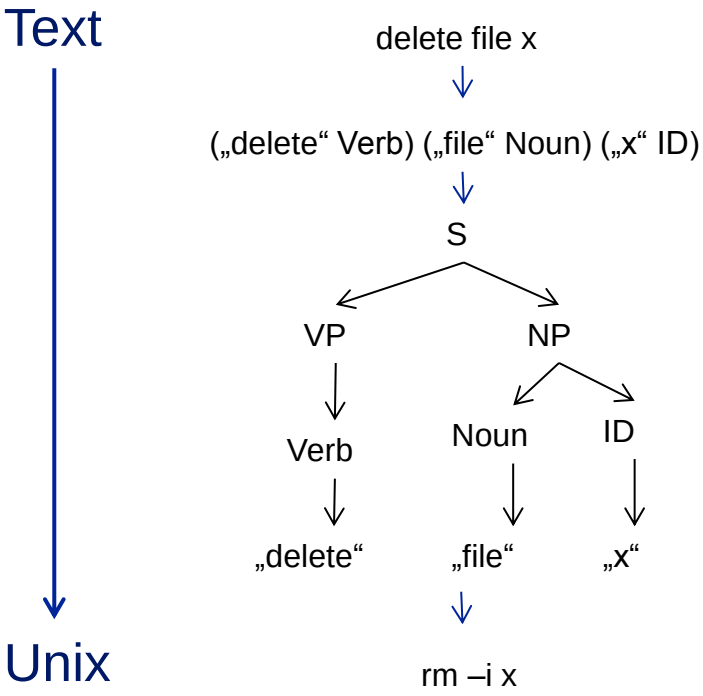


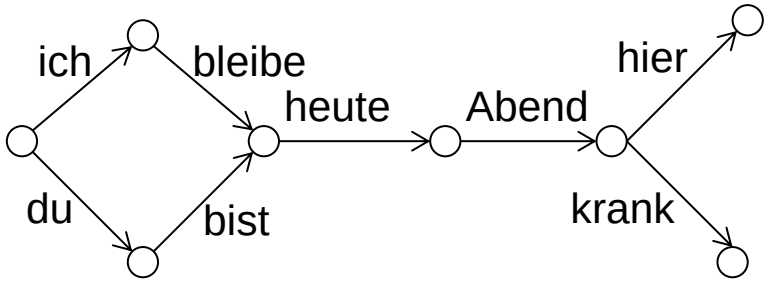
Statistische Sprachmodelle
Wozu?

- Übersetzung:
  - Übersetzungsmodell + Sprachmodell der Zielsprache.

Übersetzungsmodell

Sprachmodell

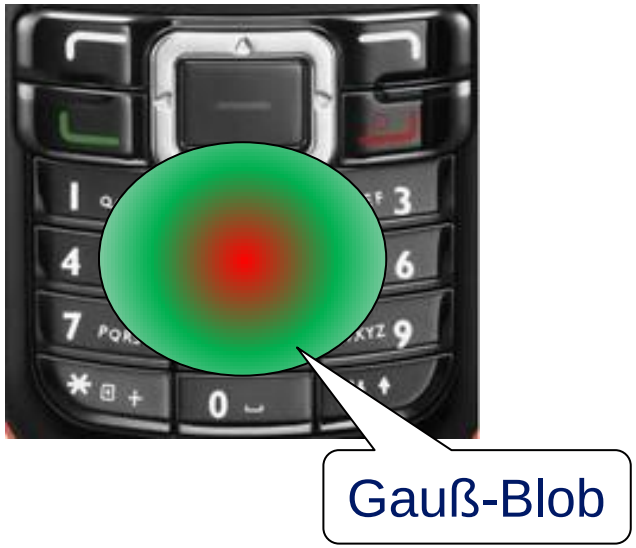




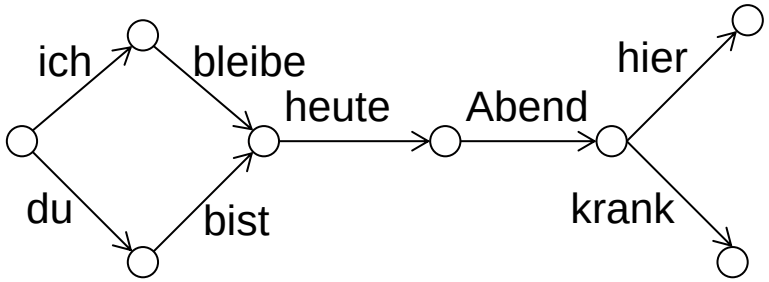
Statistische Sprachmodelle
Wozu?

- Smartphone-Tastatureingabe :
  - Normalverteilte Fingerposition + Sprachmodell.

Normalverteilte Fingerposition



Sprachmodell



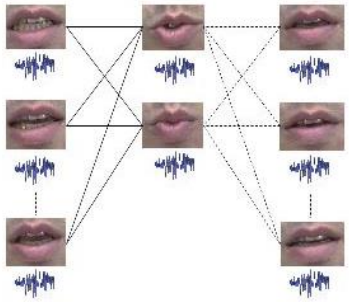
Statistische Sprachmodelle
Die Formel

- Satz von Bayes („DIE Formel“):

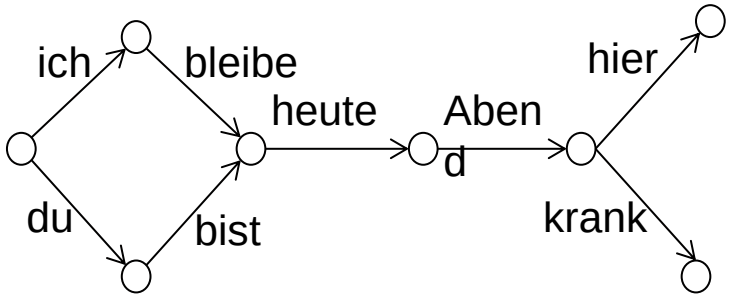
$$\arg\max_{w_1,\ldots,w_T} P(w_1,\ldots,w_T \mid signal)$$

$$= \arg\max_{w_1,\ldots,w_T} \underbrace{P(signal \mid w_1,\ldots,w_T)}_{\text{Likelihood}} \; \underbrace{P(w_1,\ldots,w_T)}_{\text{Prior}}$$

Spracherkennung:	Akustikmodell (HMM)	Sprachmodell auf Wortebene
Handschrifterkennung:	Schriftmodell	Sprachmodell auf Wortebene
Rechtschreibkorrektur:	Vertipp-/Fehlermodell	Sprachmodell auf Wortebene
Übersetzung:	Übersetzungsmodell	Sprachmodell auf Wortebene
PDA-Tastatureingabe:	Stiftpositionsmodell	Sprachmodell auf Buchstabenebene
T9	Buchst.-Tastenzuordnung	Sprachmodell auf Buchstabenebene



X



N-Gramm-Modelle

- Wahrscheinlichkeit für einen gegebenen Satz:
  - P(„ich pflücke Beeren im Wald“) = P(„ich“) x P(„pflücke“ | „ich“) x
    P(„Beeren“ | „ich pflücke“) x
    P(„im“ | „ich pflücke Beeren“) x
    P(„Wald“ | „ich pflücke Beeren im“).
- Problem: Zu viele Wahrscheinlichkeiten.
- Markov-Annahme (N-1). Ordnung (bsp: 2. Ordnung):
  - P(„Wald“ | „ich pflücke Beeren im“) = P(„Wald“ | „Beeren im“).

N-Gramm-Modelle

- Markov-Annahme (N-1). Ordnung (bsp: 2. Ordnung):
  - P(„Wald“ | „ich pflücke Beeren im“) = P(„Wald“ | „Beeren im“).

- Modell speichert Wahrscheinlichkeiten für Wortfolgen der Länge maximal N.
- N=1: „Unigramm“,
- N=2: „Bigramm“,
- N=3: „Trigramm“.

N-Gramm Modelle
Vorhersage

- Vorhersage des nächsten Wortes:
  - Bestimme $P(w_t \mid w_1, ..., w_{t-1})$.
  - Die meisten Wortfolgen $w_1, ..., w_t$ tauchen in einem Korpus nur einmal auf,
  - wie können wir dafür Wahrscheinlichkeiten ermitteln?
- Markov-Annahme (N-1)-ter Ordnung:
  - Nur die letzten N-1 Wörter sind entscheidend für das nächste (N-te) Wort.
  - $P(w_{t+N} \mid w_{t+N-1}, ..., w_1) = P(w_{t+N} \mid w_{t+N-1}, ..., w_t)$

Ignoriere Wörter w_{t-1} bis w_1

N-Gramm-Modelle
Markov-Annahme

- Wahrscheinlichkeit für Wort $t+N$ ergibt sich aus Wörtern t bis $t+N-1$.
  - „ich pflücke heute nachmittag wieder ...“ [Beeren],

    N = 4

    N = 5
  - „ich jage heute nachmittag wieder ...“ [Bären]

    N = 4

    N = 5
  - $N \leq 4$: Etwa gleiche Wahrscheinlichkeiten für nächstes Wort;
  - $N \geq 5$: Unterschiedliche Wahrscheinlichkeiten.

N-Gramm-Modelle
Wahl von N

- Großes N: Wenige Beispiele für jedes N-Gramm; Schätzen der Wahrscheinlichkeiten schwierig.
  - Vorteil: Wenig Kontextinformation geht verloren.
  - Nachteil: Viele Wahrscheinlichkeiten müssen geschätzt werden (viele 0).

- Kleines N: Viele Beispiele, aber Vorhersage ungenau da verschiedene Wortfolgen als gleich behandelt werden.
  - Vorteile: Wenige Wortfolgen, die häufig vorkommen. Vorhersage stützt sich auf viele Beispiele.
  - Nachteil: Kontextinformation geht verloren.

N-Gramm-Modelle
Wahl von N

- Wie viele Wahrscheinlichkeiten müssen wir kennen?
  - N=1 (nullte Ordnung): ca. 20,000 verschiedene Wortfolgen der Länge 1,
  - N=2 (erste Ordnung): ca. 400,000,000,
  - N=3 (zweite Ordnung): ca 8×10^{12}.
  - N=4 (dritte Ordnung): ca 1.6×10^{17}.

- N-Gramme über Wortstämme, statt über Wörter.
  - Etwas Information geht verloren, aber deutlich weniger Wahrscheinlichkeiten müssen geschätzt werden.
  - Beispiel: Kategorie → kategor
    Kategorien → kategor

N-Gramm-Modelle
Lernen, Multinomialverteilung

- Verteilung über m verschiedene Belegungen einer Zufallsvariable
- bestimmt durch Parametervektor $\boldsymbol{\theta}=(\theta_1,\ldots,\theta_m)$
  - mit der Eigenschaft $\theta_i \geq 0,\ \Sigma_i\ \theta_i = 1$
- Wahrscheinlichkeitsverteilung:

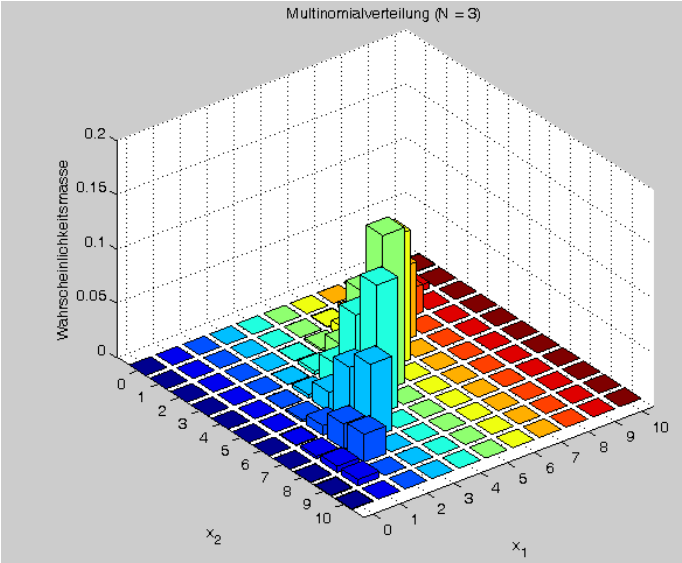
$$P(x_1,\ldots,x_n) = \underbrace{\binom{N}{x_1,\ldots,x_n}}_{\text{Multinomialkoeffizient}} \theta_1^{x_1} \cdots \theta_m^{x_n}$$

Multinomialverteilung
Beispiel

$$P(x_1,\ldots,x_n) = \binom{N}{x_1,\ldots,x_n} \theta_1^{x_1} \cdots \theta_m^{x_n}$$

- Bei einem Roulett-Spiel gibt es 38 Verschiedene Möglichkeiten: 18 mal rot, 18 mal schwarz, 2 mal grün.
  - Daraus ergeben sich folgende Parameter der Multinomialverteilung: $\theta_1 = \theta_2 = \frac{18}{38}, \theta_3 = \frac{2}{38}$.
  - Frage: Was ist die Wahrscheinlichkeit für 4 mal rot, 2 mal schwarz und 4 mal grün?
  - Antwort:

$$P(4,2,4) = \left(\frac{10!}{4!2!4!}\right)\left(\frac{18}{38}\right)^4\left(\frac{18}{38}\right)^2\left(\frac{2}{38}\right)^4$$



N-Gramm-Modelle
Lernen, Dirichletverteilung

- Konjugierter Prior zur Multinomialverteilung.
- Verteilung über m-dimensionale Vektoren mit reellwertigen Einträgen in $[0;1]$ mit Summe der Einträge = 1.
- Parametrisiert durch m-dimensionalen Vektor $\boldsymbol{\alpha}$ mit positiven Einträgen.
- Dichtefunktion:

$$P(\theta_1, \ldots, \theta_m | \alpha_1, \ldots, \alpha_m) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{m} \theta_i^{\alpha_i - 1}$$

verallgemeinerte Beta-Funktion

  - Spezialfall bei m=2: Beta-Verteilung

N-Gramm-Modelle
Lernen

- Korpus D ist Folge von Zufallsvariablen w_i; Wertebereich ist das Vokabular v_1 bis v_K.
- Beobachtungen:
  - $x_{v_1 \ldots v_1}, \ldots, x_{v_K \ldots v_K}$: Anzahl der Vorkommen der n-Gramme $v_1 \ldots v_1$ bis $v_K \ldots v_K$ in D.
- Parameter $\boldsymbol{\theta} = (\theta_{v_1 | \ldots v_1}, \ldots, \theta_{v_K | \ldots v_K})$

$$\theta_{v_{i_n} | \ldots v_{i_1}} = P(w_n = v_{i_n} \mid w_{n-1} \ldots w_1 = v_{i_{n-1}} \ldots v_{i_1}, \boldsymbol{\theta})$$

- Wie viele Parameter gibt es?

N-Gramm-Modelle
Wahrscheinlichkeit eines Satzes

- Wahrscheinlichkeit des Satzes $w_1,\ldots,w_T$ bei gegebenem N-Gramm-Modell:
  - $$\begin{aligned} P(w_1,\ldots,w_T) &= P(w_1)P(w_2 \mid w_1)\ldots P(w_T \mid w_{T-1},\ldots,w_1) \\ &= P(w_1)P(w_2 \mid w_1)\ldots P(w_T \mid w_{T-1},\ldots,w_{T-n}) \\ &= \prod_{i=1}^{N-1} P(w_i \mid w_{i-1},\ldots,w_1) \prod_{i=N}^{T} P(w_i \mid w_{i-1},\ldots,w_{i-n}) \\ &= \theta_{w_1} \prod_{i=2}^{N-1} \theta_{w_i \mid \ldots w_1} \prod_{i=N}^{T} \theta_{w_i \mid \ldots w_{i-n}} \end{aligned}$$

Markov-Eigenschaft

- Beispiel: 3-Gramm-Modell:
  - $$\begin{aligned} P(w_1,\ldots,w_T) &= P(w_1)P(w_2 \mid w_1)\ldots P(w_T \mid w_{T-1},\ldots,w_1) \\ &= P(w_1)P(w_2 \mid w_1) \prod_{i=3}^{T} P(w_i \mid w_{i-1}, w_{i-2}) \\ &= \theta_{w_1} \theta_{w_2 \mid w_1} \prod_{i=3}^{T} \theta_{w_i \mid w_{i-1} w_{i-2}} \end{aligned}$$

N-Gramm-Modelle
Empirische Inferenz der Modellparameter

- Parameter des Modells sind die echten Auftretenswahrscheinlichkeiten $\boldsymbol{\theta} = (\theta_{v_1|\ldots v_1}, \ldots, \theta_{v_K|\ldots v_K})$.
- Wahrscheinlichkeiten sind nicht bekannt.
- Echte Wahrscheinlichkeiten sind in einem Korpus von Trainingsdokumenten reflektiert.

- Empirische Inferenz: Berechnung des Posteriors
  - $P(\boldsymbol{\theta} \mid D) \propto P(D \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})$
- MAP-Hypothese: Wahrscheinlichste Parameter gegeben das Korpus D.

N-Gramm-Modelle
Inferenz: Maximum-Likelihood

- Posterior:
  - $P(\boldsymbol{\theta} \mid D) \propto P(D \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})$
- Likelihood: Produkt von Multinomialverteilungen.
  - $$P(D \mid \boldsymbol{\theta}) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}, \ldots, w_{t-n+1}, \boldsymbol{\theta})$$
    $$= \prod_{t=1}^{T} \theta_{w_t \mid w_{t-1}, \ldots, w_{t-n+1}}$$

    Laut Definition der Multinomialverteilung
- Maximum-Likelihood- (ML-)Schätzer:
  - $$\boldsymbol{\theta}^{ML} = \arg\max P(D \mid \boldsymbol{\theta}) = \arg\max P\left(D \mid \begin{pmatrix} \theta_{v_1 \mid \ldots v_1} \\ \ldots \\ \theta_{v_K \mid \ldots v_K} \end{pmatrix} \right)$$

    Auftrittshäufigkeit von $v_{i_n} \ldots v_{i_1}$

    $$\theta^{ML}_{v_{i_n} \mid \ldots v_{i_1}} = \frac{x_{v_{i_n} \ldots v_{i_1}}}{x_{v_{i_{n-1}} \ldots v_{i_1}}}$$

N-Gramm-Modelle
Inferenz: Maximum A-Posteriori

- Posterior:
  - $P(\boldsymbol{\theta} \mid D) \propto P(D \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})$
- Prior: Dirichlet, konjugiert zur Multinomialverteilung.
  - $P(\theta) = \prod_{w_{n-1} \ldots w_1 = v_1 \ldots v_1}^{v_K \ldots v_K} P_{Dirichlet}(\theta_{v_1 \mid w_{n-1} \ldots w_1}, \ldots, \theta_{v_K \mid w_{n-1} \ldots w_1} \mid \alpha_{v_1}, \ldots, \alpha_{v_K})$
- Posterior: Wieder Dirichletverteilt.

$$
\begin{aligned}
P(\theta \mid D) &= \frac{1}{P(D)} \overbrace{\prod_{t=1}^{T} \theta_{w_t \mid w_{t-1}, \ldots, w_{t-n+1}}}^{\text{Likelihood}} \overbrace{\prod_{w_{n-1} \ldots w_1 = v_1 \ldots v_1}^{v_K \ldots v_K} P_{Dirichlet}(\theta_{v_1 \mid w_{n-1} \ldots w_1}, \ldots, \theta_{v_K \mid w_{n-1} \ldots w_1} \mid \alpha_{v_1}, \ldots, \alpha_{v_K})}^{\text{Prior}} \\
&= \prod_{w_{n-1} \ldots w_1 = v_1 \ldots v_1}^{v_K \ldots v_K} P_{Dirichlet}\left(\theta_{v_1 \mid w_{n-1} \ldots w_1}, \ldots, \theta_{v_K \mid w_{n-1} \ldots w_1} \mid \alpha_{v_1} + x_{v_1 w_{n-1} \ldots w_1}, \ldots, \alpha_{v_K} + x_{v_K w_{n-1} \ldots w_1}\right)
\end{aligned}
$$

N-Gramm-Modelle
Inferenz: Maximum A-Posteriori

- Posterior: wieder Dirichletverteilt.
  - $P(\boldsymbol{\theta} \mid D)$

$$= \prod_{w_{n-1}\ldots w_n = v_1 \ldots v_1}^{v_K \ldots v_K} P_{Dirichlet}\left(\theta_{v_1 \mid w_{n-1}\ldots w_1}, \ldots, \theta_{v_K \mid w_{n-1}\ldots w_1} \mid \alpha_{v_1} + x_{v_1 w_{n-1}\ldots w_1}, \ldots, \alpha_{v_K} + x_{v_K w_{n-1}\ldots w_1}\right)$$

- MAP-Hypothese:
  - $\arg\max_{\theta_{v_{i_n} \mid w_{n-1}\ldots w_1}} P(\boldsymbol{\theta} \mid D)$

$$= \frac{x_{v_{i_n} w_{n-1}\ldots w_1} + \alpha_{v_{i_n}} - 1}{x_{w_{n-1}\ldots w_1} + \sum_{j=1}^{K} (\alpha_{v_j} - 1)}$$

α=2: „Laplace Smoothing“

- Sonderbehandlung $n<N$: Wahrscheinlichkeiten nur aus Satzanfängen schätzen.

N-Gramm-Modelle
Inferenz der Hyperparameter

- Einstellen der Hyperparameter α.
- α: Wahrscheinlichkeit für ein N-Gramm, das im Trainingskorpus nicht auftaucht:
  - $\theta_{v_{i_n}|w_{n-1}\ldots w_1} = \dfrac{0+\alpha_{v_{i_n}}-1}{x_{w_{n-1}\ldots w_1}+\sum_{j=1}^{K}(\alpha_{v_j}-1)}$;
  - Bei einem Wert für alle α: $\dfrac{\alpha-1}{x_{w_{n-1}\ldots w_1}+(\alpha-1)K}$.

N-Gramm-Modelle
Inferenz der Hyperparameter

- Wie viel Wahrscheinlichkeitsmasse sollte auf ungesehene Kombinationen entfallen?
- Teile Korpus in zwei Teile.
  - Zähle, wie viele Kombinationen $w_n \mid w_{n-1} \ldots w_1$ im zweiten Teil, aber nicht im ersten Teil auftauchen.
  - Stelle dann α so ein, dass $\frac{\alpha - 1}{x_{w_{n-1} \ldots w_1} + (\alpha - 1)K}$ im Durchschnitt über alle Kontexte $w_{n-1} \ldots w_1$ dem gewünschten Verhältnis entspricht.
- Kleine Schwäche des Verfahrens: Im ersten Teil des Korpus kommen mehr N-Gramme *nicht* vor als im gesamten Korpus. (Wahrscheinlichkeit für ungesehene Kombinationen wird überschätzt)

N-Gramm-Modelle
Einstellen von N (Markov-Grad)

- N ist entscheidender Parameter des N-Gramm-Modells.
- Großes N : viele Wahrscheinlichkeiten müssen geschätzt werden.
- Kleines N : zu viel Kontextinformation geht verloren.
- Kompromiss: Interpolation.

N-Gramm-Modelle
Interpolation

- Dem N-Gramm-Sprachmodell liegt ein generatives Prozessmodell zugrunde.
  - Wörter werden iterativ „ausgewürfelt“.
  - Jedes Wort w_i wird nach der Wahrscheinlichkeit $P(w_i \mid w_{i-1},\ldots,w_{i-n+1})$ ausgewürfelt.

- Generatives Modell für interpoliertes N-Gramm:
  - Für jedes Wort w_i wird zunächst Markov-Grad n zwischen 1 und N nach $p(n)$ gezogen.
  - Wort w_i wird dann nach $P(w_i \mid w_{i-1},\ldots,w_{i-n+1})$ gezogen.

Interpoliertes N-Gramm-Modell
Wahrscheinlichkeit eines Satzes

- Wahrscheinlichkeit des Satzes $w_1, \ldots, w_T$ mit interpoliertem N-Gramm-Modell:
  - $$\begin{aligned} P(w_1,\ldots,w_T) &= P(w_1)P(w_2 \mid w_1)\ldots P(w_T \mid w_{T-1},\ldots,w_1) \\ &= P(w_1)(p(1)P(w_2) + p(2)P(w_2 \mid w_1))\ldots\sum_{n=1}^{N} p(n)P(w_T \mid w_{T-1},\ldots,w_{T-n+1}) \\ &= p(w_1)\prod_{i=1}^{N-1}\sum_{n=2}^{i} p_i(n)P(w_i \mid w_{i-1},\ldots,w_1)\prod_{i=N}^{T}\sum_{n=1}^{N} p_i(n)P(w_i \mid w_{i-1},\ldots,w_{i-n+1}) \end{aligned}$$

- Beispiel: N = 3.
  - $$\begin{aligned} &P(w_1,\ldots,w_T) \\ &= P(w_1)(p(1)P(w_2) + p(2)P(w_2 \mid w_1))\prod_{i=3}^{T}\sum_{n=1}^{3} p(n)P(w_i \mid w_{i-1}, w_{i-2}) \end{aligned}$$

Interpoliertes N-Gramm-Modell
Inferenz der Parameter

- Likelihood für interpoliertes N-Gramm:
  - $$P(D \mid \boldsymbol{\theta}) = \prod_{t=1}^{T} \sum_{n_{t=1}}^{N} P(n_t) P(w_t \mid w_{t-1}, \ldots, w_{t-n_t+1}, \boldsymbol{\theta})$$
    $$= \prod_{t=1}^{T} \sum_{n_{t=1}}^{N} p(n_t) \theta_{w_t \mid w_{t-1}, \ldots, w_{t-n_t+1}}$$
- Posterior:
  - $$P(\boldsymbol{\theta} \mid D)$$
    $$= \prod_{t=1}^{T} \sum_{n_t=1}^{N} p(n_t) \theta_{w_t \mid w_{t-1}, \ldots, w_{t-n_t+1}} \prod_{w_{n-1} \ldots w_1 = v_1 \ldots v_1}^{v_K \ldots v_K} \sum_{n=1}^{N} p(n) P_{Dirichlet}(\theta_{v_1 \mid w_{n-1} \ldots w_1}, \ldots, \theta_{v_K \mid w_{n-1} \ldots w_1} \mid \alpha_{v_1}, \ldots, \alpha_{v_K})$$
    $$= \prod_{w_{n-1} \ldots w_n = v_1 \ldots v_1}^{v_K \ldots v_K} \sum_{n=1}^{N} p(n) P_{Dirichlet}\left(\theta_{v_1 \mid w_{n-1} \ldots w_1}, \ldots, \theta_{v_K \mid w_{n-1} \ldots w_1} \mid \alpha_{v_1} + x_{v_1 w_{n-1} \ldots w_1}, \ldots, \alpha_{v_K} + x_{v_K w_{n-1} \ldots w_1}\right)$$
- Problem beim Parameterschätzen:
  - n_t ist latente, unbeobachtete Variable.

Interpoliertes N-Gramm-Modell
Parameterschätzung

- Beim Schätzen der Parameter der N-Gramm-Modelle müssten wir wissen, welches Wort mit welchem Abhängigkeitswert N erzeugt wurde.
- Nur die Wörter, die mit Wert N generiert wurden sollten in Schätzung des N-Gramm-Modells eingehen.
- Definition Zufallsvariable z_{in}: Wert $z_{in}=1$ zeigt an, dass Wort w_i mit Abhängigkeit N gezogen wurde.
- Problem: Werte der z_{in} sind aber unbekannt.

Interpoliertes N-Gramm-Modell
Parameterschätzung

- Problem: Schätzproblem (Schätzen der Paramater $\boldsymbol{\theta} = (\theta_{v_1|\ldots v_1}, \ldots, \theta_{v_K|\ldots v_K})$) mit latenten Variablen z_{in}.
- Lösung: Expactation-Maximization-Algorithmus.
- Idee: Beginne mit zufälligen Parametern.
  - Expactation-Schritt: Berechne Wahrscheinlichkeiten $p(z_{in})$ gegeben die Daten und Parameter.
  - Maximization-Schritt: Schätze $\boldsymbol{\theta} = (\theta_{v_1|\ldots v_1}, \ldots, \theta_{v_K|\ldots v_K})$ basierend auf den aktuellen Werten der $p(z_{in})$.
- Wiederhole bis zur Konvergenz.

Expactation Maximization -Algorithmus
Motivation und Beispiel

- Münzwurfexperiment mit zwei Münzen:
  - Münze A mit Parameter θ_A
  - Münze B mit Parameter θ_B

Wahrscheinlichkeit für Zahl

- Ziel: Parameter $\theta = (\theta_A, \theta_B)$ schätzen.
- Experiment:
  - Wiederhole für $t = 1, \dots, 5$:
    - Wähle zufällig eine Münze $z_t = \{A, B\}$ und wirf die Münze 10 mal ($x_t \in \{K, Z\}^{10}$).
- Parameterschätzung (ML):
  - $\hat{\theta}_A = \frac{\#\ \text{Würfe mit Kopf für Münze } A}{\#\ \text{Würfe mit Münze } A}$

Analog für $\hat{\theta}_B$

- Problem: Wie Parameter schätzen, wenn die Zufallsvariablen z_t unbekannt sind?

Expactation Maximization
Basisalgorithmus

- Idee: Wenn wir wissen, welche Münze wann geworfen wurde, ist Parameterinferenz einfach (siehe letzte Folie).

$$\hat{\theta}_A = \frac{\#\ \text{Würfe mit Kopf für Münze } A}{\#\ \text{Würfe mit Münze } A}$$

- Algorithmus (für Münzwurfexperiment):
  - 1. Starte mit initialen Parametern $\hat{\theta}_A$und $\hat{\theta}_B$.
  - 2. Expactation: Bestimme für alle $t = 1, \dots, 5$ Münzwurfserien, mit welcher Wahrscheinlichkeit z_t die Münzen die Serien produziert haben.
  - 3. Maximization: Nutze $z_1, \dots, z_5$, um die optimalen Parameter $\hat{\theta}_A$und $\hat{\theta}_B$ zu bestimmen.
  - 4. Wiederhole 2. und 3. bis zur Konvergenz.

Expactation Maximization
Beispiel

z_1	K	Z	Z	Z	K	K	Z	K	Z	K
z_2	K	K	K	K	Z	K	K	K	K	K
z_3	K	Z	K	K	K	K	K	Z	K	K
z_4	K	Z	K	Z	Z	Z	K	K	Z	Z
z_5	Z	K	K	K	Z	K	K	K	Z	K

unbekannt

$$\frac{(0.5^{10})}{(0.5^{10})+(0.6^5*0.4^5)}$$

0.45 x A	0.55 x B
0.8 x A	0.2 x B
0.73 x A	0.27 x B
0.35 x A	0.65 x B
0.65 x A	0.35 x B

$0.45 * 5$

Münze A	Münze B
2.2 K, 2.2 Z	2.8 K, 2.8 Z
7.2 K, 0.8 Z	1.8 K, 0.2 Z
5.9 K, 1.5 Z	2.1 K, 0.5 Z
1.4 K, 2.1 Z	2.6 K, 3.9 Z
4.5 K, 1.9 Z	2.5 K, 1.1 Z
Σ 21.3 K, 8.6 Z	11.7 k, 8.4 Z

Expactation

Maximization

$$\hat{\theta}_A^{(1)} = \frac{21.3}{21.3+8.6} \approx 0.71$$
$$\hat{\theta}_B^{(1)} = \frac{11.7}{11.7+8.4} \approx 0.58$$

$$\hat{\theta}_A^{(0)} = 0.6$$
$$\hat{\theta}_B^{(0)} = 0.5$$

Initialisierung

$$\hat{\theta}_A^{(10)} \approx 0.8$$
$$\hat{\theta}_B^{(10)} \approx 0.52$$

Interpoliertes N-Gramm-Modell
Expectation-Schritt

- $p(z_{in})$: Wahrscheinlichkeit dafür, dass w_i von n-Gramm-Modell generiert wurde:
  - $p(z_{in} = 1) = p(n \mid w_i) \propto p(w_i \mid w_{i-1}, \ldots, w_{i-n+1})\, p(n)$

Interpoliertes N-Gramm-Modell
Maximization-Schritt

- MAP-Schätzer:
  - $$\arg\max_{\theta_{v_{i_n}|v_{i_{n-1}}\ldots v_{i_{n_1}}}} P(\theta \mid D) = \frac{\bar{x}_{v_{in} v_{i_{n-1}} \ldots v_{i_{n_1}}} + \alpha_{v_{in}} - 1}{\bar{x}_{v_{i_{n-1}} \ldots v_{i_{n_1}}} - \sum_{i=1}^{K} (\alpha_{v_{in}} - 1) K}$$
- Zählvariablen:
  - $$\bar{x}_{v_{i_n} v_{in-1} \ldots v_{i1}} = \sum_{t=1}^{T} p(z_{tn}) [[w_t \ldots w_{t-n} = v_{i_n} v_{in-1} \ldots v_{i1}]]$$

Indikatorfunktion

N-Gramm-Modelle
Anzahl der Parameter: Beispiel

- IBM Tangora:
  - Korpus mit 365 Mio Wörtern, (260.000 Verschiedene)
  - 6.8×10^{10} 2-Gramme, 14 Mio davon tauchen auf, 8 Mio nur einmal; 2.2% aller neuen 2-Gramme sind unbekannt.
  - 1.8×10^{16} 3-Gramme, 75 Mio tauchen auf, 53 Mio einmal, 14.7% aller neuen 3-Gramme sind unbekannt.

N-Gramm-Modelle
Evaluation

- N-Gramm auf Trainingskorpus gelernt.
- Wie gut ist es?
- Bewertung auf Testkorpus.
  - Nicht auf Trainingskorpus.
- Wie sehr überrascht jedes neue Wort des Testkorpus?
  - Gutes Modell → Weniger Überraschung.
- Entropie H: Wie viele Bit brauche ich um nächstes Wort zu kodieren?
- Perplexität („mittlerer Verzweigungsfaktor“): 2^H.
- Minimum der Entropie = Minimum der Perplexität wenn gelerntes Modell gleich „echtem“ Modell ist.

N-Gramm-Modelle
Evaluation

- N-Gramm-Modell und ein langer Text W.
- Kreuzentropie des Textes gegeben das Modell:
  - $H(W|m) = -\frac{1}{|W|}\log_2 P_m(W)$

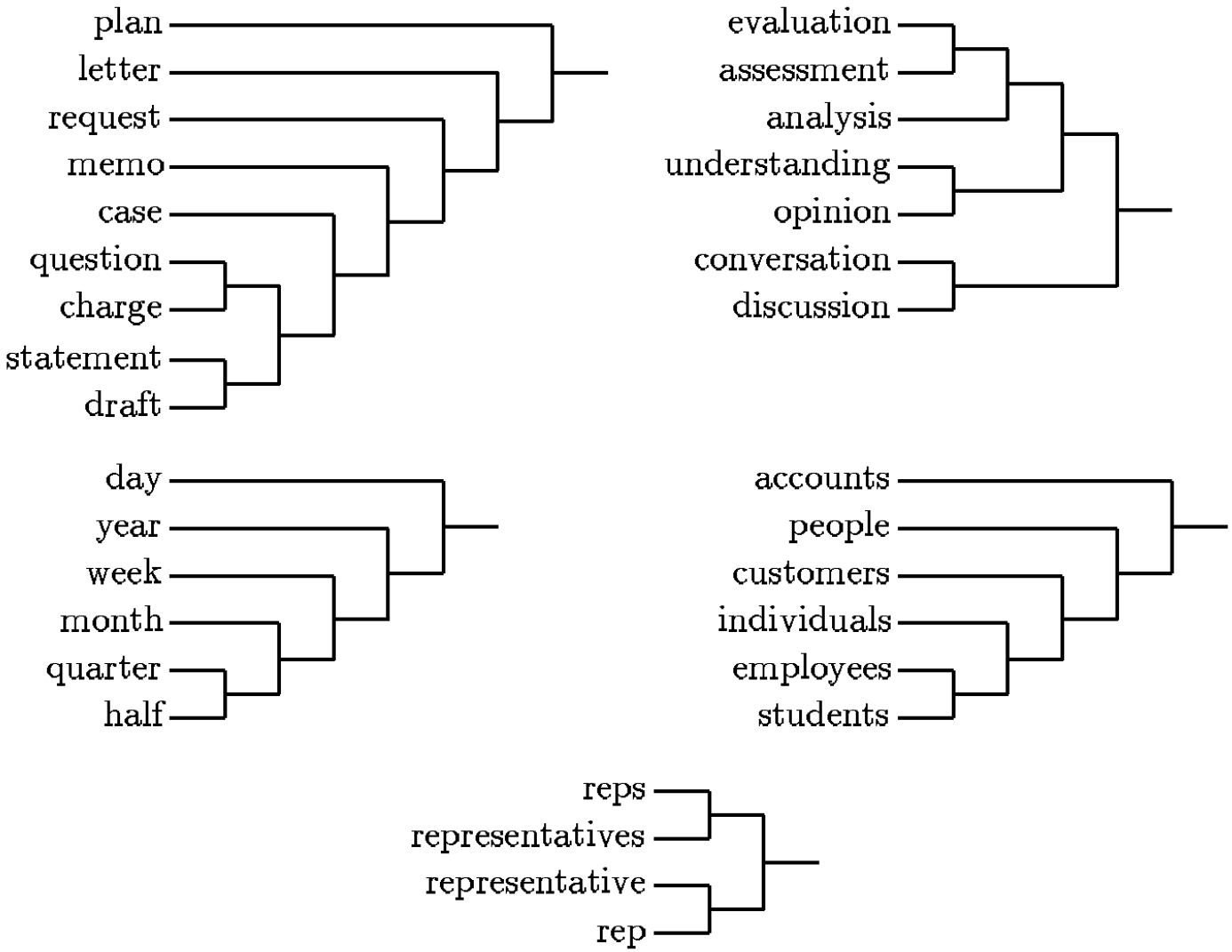
    Wahrscheinlichkeit für Wortfolge für ein Gegebenes Modell m
  - $H(W|m) = -\frac{1}{|W|}\log_2 P_m(w_1 \dots w_n) = -\frac{1}{|W|}\sum_i \log_2 P_m(w_i|w_{i-1}, \dots, w_1)$

- Perplexität des Modells (durchschnittlicher Verzweigungsfaktor):
  - $PP(W|m) = 2^{H(W|m)}$

N-Gramm-Klassenmodelle

- Manche Wörter haben denselben Kontext.
  - „Wir sehen uns [Montag | Dienstag | ...] Nachmittag“.
- Idee: Wortklassen. $\pi(w)=c$.
- Ein N-Gramm-Modell ist ein Klassenmodell gdw.
- $P(w_n \mid w_{n-1},\ldots,w_1) = P(w_n \mid c_n)P(c_n \mid c_{n-1},\ldots,c_1)$
- Klassenmodell hat weniger Parameter (wenn C < V).
- N-Gramm-Klassenmodell = N-Gramm-Modell, wenn jedes Wort eine eigene Klasse hat.
- N-Gramm-Klassenmodell mit einem Wort pro Klasse wird normal aus Trainingskorpus gelernt.
- Dann werden Klassen zusammen gelegt.

N-Gramm-Klassenmodelle



- Beispiel-Dendrogramme

Beispielaufteilung in 1000 Klassen

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays

June March July April January December October November September August

people guys folks fellows CEOs chaps doubters commies unfortunates blokes

down backwards ashore sideways southward northward overboard aloft downwards adrift

water gas coal liquid acid sand carbon steam shale iron

great big vast sudden mere sheer gigantic lifelong scant colossal

man woman boy girl lawyer doctor guy farmer teacher citizen

American Indian European Japanese German African Catholic Israeli Italian Arab

pressure temperature permeability density porosity stress velocity viscosity gravity tension

mother wife father son husband brother daughter sister boss uncle

machine device controller processor CPU printer spindle subsystem compiler plotter

John George James Bob Robert Paul William Jim David Mike

- Beispielklassen

Anwendung von N-Grammen
Rechtschreibkorrektur

- Gegeben: Signal $s_1, \ldots, s_T$; gesucht: Wahrscheinlichste beabsichtigte Abfolge $w_1, \ldots, w_T$.
- Fehlermodell: $p(s_i \mid w_i)$.
  - Z.B.: $p(s_i \mid w_i) = \exp\left(-\gamma_1 d_{edit}(s_i, w_i) - \gamma_2 d_{phonetisch}(s_i, w_i) - \gamma_3 d_{keyboard}(s_i, w_i)\right)$
- Dekodierung:
  - $$\begin{aligned} &\arg\max_{w_1,\ldots,w_T} p(w_1,\ldots,w_T \mid s_1,\ldots,s_T) \\ &= \arg\max_{w_1,\ldots,w_T} p(s_1,\ldots,s_T \mid w_1,\ldots,w_T)\, p(w_1,\ldots,w_T) \\ &= \arg\max_{w_1,\ldots,w_T} \prod_{i=1}^{T} p(s_i \mid w_i)\, p(w_i \mid w_{i-1},\ldots,w_{i-n}) \end{aligned}$$

Satz von Bayes

Unabhängigkeits-annahme + Markov Annahme

Anwendung von N-Grammen
Rechtschreibkorrektur

- Dekodierung:
  - $$\arg\max_{w_1,\ldots,w_T} p(w_1,\ldots,w_T \mid s_1,\ldots,s_T)$$
    $$= \arg\max_{w_1,\ldots,w_T} \prod_{i=1}^{T} p(s_i \mid w_i) p(w_i \mid w_{i-1},\ldots,w_{i-n})$$
- Naive Implementierung:
  - Suche über alle $w_1, \ldots, w_T$ in $O(V^T)$.
- Mit dynamischer Programmierung:
  - Suche über alle $w_1, \ldots, w_T$ in $O(TV^n)$.
  - Idee: Tabelle mit $p(w_n \mid w_1, \ldots, w_{n-1})$ in $O(V^n)$ vorberechnen.
  - Dann linear durch Signalabfolge gehen, für jedes n-Gramm nur die wahrscheinlichste Historie merken

Anwendung von N-Grammen
Rechtschreibkorrektur

- Dekodierung:
  - $$\arg\max_{w_1,\ldots,w_T} p(w_1,\ldots,w_T \mid s_1,\ldots,s_T)$$
    $$= \arg\max_{w_1,\ldots,w_T} \prod_{i=1}^{T} p(s_i \mid w_i) p(w_i \mid w_{i-1},\ldots,w_{i-n})$$
- Weitere Optimierung:
  - Für jedes s_i, nur die k besten w_i in Erwägung ziehen.
  - Suche in $O(Tk^n)$.
- Optimierung für Suchmaschinen:
  - Für häufig falsch geschriebene Suchanfragen (z.B. „Gadaffi“) wahrscheinlichste Bedeutung in Cache vorhalten.

Anwendung von N-Grammen
Auto-Complete

- Dekodierung:
  - $$\arg\max_{w_{T+k},\ldots,w_{T+1}} p(w_1,\ldots,w_T \mid \boldsymbol{\theta})$$
    $$= \arg\max_{w_{T+k},\ldots,w_{T+1}} \prod_{i=1}^{T} p(w_{T+i} \mid w_{T+i-1},\ldots,w_{T+i-n})$$
- Suche mit dynamischer Programmierung:
  - $O(TV^n)$.

Zusammenfassung

- Sprachmodelle bestimmen Wahrscheinlichkeit für Elemente (z.B. Sätze) einer Sprache.
- N-Gramme: Markov Annahme (N-1). Ordnung.
  - Lernen der Parameter: Empirsche Inferenz der Modellparameter (Wahrscheinlichkeiten für N-Gramme).
- Erweiterung: Interpoliertes N-Gramm-Modell.
  - Lernen der Parameter: EM-Algorithmus.
  - Wahrscheinlichkeiten für N-Gramme + Wahrscheinlichkeiten für ein bestimmtes N.

Fragen?