Sprachsynthese: Graphem-Phonem-Konvertierung

Uwe Reichel
Institut für Phonetik und Sprachverarbeitung
Ludwig-Maximilians-Universität München
reichelu@phonetik.uni-muenchen.de

2. Dezember 2014

Inhalt

- Einflussfaktoren
- Alinierung
- Konvertierung
  - Table Lookup with Defaults (van den Bosch et al., 1993)
  - Maschinelles Lernen
  - Entscheidungsbäume
- Silbifizierung
  - in Graphemfolge
  - in Phonemfolge
- Wortbetonung

Einflussfaktoren

Notation: *Grapheme in spitzen Klammern; Phoneme in Slashes nach German Sampa;*

Kleinbuchstaben - Grapheme; Großbuchstaben - Graphemvariablen (V - Vokal)

- Ist die Abbildung $<Graphem> \longrightarrow /Phonem/$ eindeutig?
- **Nein**. Beispiel $<s>$:

  $<s> \longrightarrow /s/$ *(was)*
  $<s> \longrightarrow /z/$ *(Vase)*
  $<s> \longrightarrow /S/$ *(stehen)*
  $<s> \longrightarrow /\_/$ *(Wasser)*

- weiteres Beispiel: $<u>$ in *B**u**nd* und *Q**u**alle*

Einflussfaktoren

Graphemkontext

<s> $\longrightarrow$ /z/ | *V _V* ; **aber:** *losen* **vs.** *Loserwerb*
<s> $\longrightarrow$ /S/ | _<t>; **aber:** *Stabilität vs. Rost*

. . . reicht nicht aus

Silbenstruktur

- Auslautverhärtung, keine Beeinflussung durch Graphemumgebung über Silbengrenzen hinweg
- <*g*> $\longrightarrow$ /*g* | *k*/: *Wege*, *Weg*
- <*s*> $\longrightarrow$ /*z* | *s* | *S*/: *Vase*, *Häuschen*

Einflussfaktoren

Morphologie

- morphologischer Einfluß direkt und über die Silbenstruktur manifestiert
- **direkt:** Phonem-Identität abhängig von Morphemklasse
- **Beispiele:**
  - <*er*> in *Erlöser* (/?E6l2:z6/); morph. $er_{prefix} + lös_{verb} + er_{suffix}$.
    <*er*> $\longrightarrow$ /?E6/ | im Präfix
    <*er*> $\longrightarrow$ /6/ | im Suffix
  - <*e*> in *geben* (/ge:b@n/); morph. $geb_{verb} + en_{infl}$.
    <*e*> $\longrightarrow$ /e:/ | im einsilbigen Verbstamm
    <*e*> $\longrightarrow$ /@/ | in Flektionsendung

Einflussfaktoren

- **indirekt:** morphologische Struktur bestimmt Silbenstruktur und damit Phonem-Identität
- **Beispiele:**
  - <*ng*> in *Angel* (/?aN@l/) vs. *Angelegenheit* (/?ang@le:g@nhaIt/):
    morph. $angel_{noun}$ vs. $an_{prefix}+ge_{prefix}+leg_{verb}+en_{suffix}+heit_{suffix}$.
    <*ng*> $\longrightarrow$ /N/ | keine Silbengrenze
    <*ng*> $\longrightarrow$ /n.g/ | (hier: morphologisch bedingte) Silbengrenze
  - <*se*> in *losen* (/lo:z@n/) vs. *Losentscheid* (/lo:s?EntSaIt/):
    morph. $los_{verb}+en_{infl}$ vs. $los_{noun}+ent_{prefix}+scheid_{verb}$.
    <*se*> $\longrightarrow$ /z@/ | keine Silbengrenze
    <*se*> $\longrightarrow$ /s.?E/ | (hier: morphologisch bedingte) Silbengrenze

Alinierung: Grundlagen

- Zur Gewinnung des G2P-Trainingsmaterials
- Zuordnung zusammengehöriger Abschnitte in Graphem- und Phonemsequenzen

O	b	e	r	s	c	h	u	l	z	e	u	g	n	i	s
?+o:	b	6	_	S	_	_	u:	l	t+s	OY	_	k	n	I	s

- Alinierungsproblem formuliert als **Minimierung der Distanz zwischen Graphem- und Phonemsequenz**

Alinierung: Levenshtein-Distanz

Levenshtein-Distanz

- **Minimal nötige Editierkosten** um Sequenz v (Grapheme) in Sequenz w (Phoneme) umzuwandeln
- **Standard-Editieroperationen:**
  - **Substitution** von v_i durch w_j: $<u> \longrightarrow /u:/$
  - **Löschung** von v_i: $<r> \longrightarrow \_$
  - **Einfügung** von w_j: $\_ \longrightarrow /?/$

Alinierung: Levenshtein-Distanz

Berechnung mittels dynamischer Programmierung

- **dynamische Programmierung:** sukzessives Auffüllen einer Tabelle in Abhängigkeit der bislang gegebenen Tabellenwerte
- **Beispiel: Viterbi-Algorithmus**, *siehe POS-Folien*
- **Initialisierung:** Füllen der ersten Zeile und/oder Spalte der Tabelle
- **Induktion:**
  - *informell:* wenn du etwas für $n-1$ getan/bewiesen hast, kannst du es auch für n tun/beweisen.
  - *meint hier:* sukzessives Auffüllen der restlichen Tabelle nach einem gleichbleibenden Schema.

Alinierung: Levenshtein-Distanz

Initialisierung der Tabelle:

$D[0,0] := 0$
for i:=1 **to** m: $D[i,0] := D[i-1,0] + c(v_i, \_)$
for j:=1 **to** n: $D[0,j] := D[0,j-1] + c(\_, w_j)$

Induktive Berechnung der restlichen Tabellenwerte

for i:=1 **to** m
  for j:=1 **to** n

$$D[i,j] := \min\Big[D[i-1,j-1] + c(v_i, w_j), D[i,j-1] + c(\_, w_j), D[i-1,j] + c(v_i, \_)\Big]$$

Alinierung: Levenshtein-Distanz

- 3 Möglichkeiten, wie $w_{1,j}$ aus $v_{1,i}$ hervorgeht:[1]
  1. editiere $v_{1,i-1}$ nach $w_{1,j-1}$ um und substituiere v_i durch w_j: $D[i-1, j-1] + c(v_i, w_j)$
  2. editiere $v_{1,i}$ nach $w_{1,j-1}$ um und füge w_j ein: $D[i, j-1] + c(\_, w_j)$
  3. editiere $v_{1,i-1}$ nach $w_{1,j}$ um und lösche v_i: $D[i-1, j] + c(v_i, \_)$

[1] $s_{1,j}$ bedeutet String-Präfix von s bis zur Stelle j.

Alinierung: Levenshtein-Distanz

Beispiel:

- $v =$ *Trauben*, $w =$ *Pflaume*
- geg. naive Kostenfunktion: 0 für Nullsubstitution ($v_i = w_j$), 1 sonst
- **Initialisierung:**

$\downarrow v_i$	$w_j \rightarrow$							
	ϵ	p	f	l	a	u	m	e
ϵ	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
t	**1**							
r	**2**							
a	**3**							
u	**4**							
b	**5**							
e	**6**							
n	**7**							

Alinierung: Levenshtein-Distanz

- **Induktion:**

$\downarrow v_i$	$w_j \rightarrow$							
	ϵ	p	f	l	a	u	m	e
ϵ	**0**	**1**	2	3	4	5	6	7
t	**1**	**1**						
r	2							
a	3							
u	4							
b	5							
e	6							
n	7							

Alinierung: Levenshtein-Distanz

- **Backtrace:** verfolge den Pfad der geringsten Kosten von rechts unten nach links oben zurück

$\downarrow v_i$	$w_j \rightarrow$							
	ϵ	p	f	l	a	u	m	e
ϵ	**0**	**1**	2	3	4	5	6	7
t	1	1	**2**	3	4	5	6	7
r	2	2	2	**3**	4	5	6	7
a	3	3	3	3	**3**	4	5	6
u	4	4	4	4	4	**3**	4	5
b	5	5	5	5	5	4	**4**	5
e	6	6	6	6	6	5	5	**4**
n	7	7	7	7	7	6	6	**5**

Alinierung: Levenshtein-Distanz

- **Alinierung:**

t		r	a	u	b	e	n
p+f	l	a	u	m	e	_	

Alinierung: Kostenfunktion

Naiv

- **Substitution**

$$c(v_i, w_j) = \begin{cases} 0 & : \quad v_i == w_j \\ 1 & : \quad \text{else.} \end{cases} \tag{1}$$

- **Löschung, Einfügung**

$$c(v_i, \_) = 1 \tag{2}$$

$$c(\_, w_j) = 1 \tag{3}$$

- brauchbar für Wortvergleiche (z.B. automatische Rechtschreibkorrektur)
- unbrauchbar für Graphem-Phonem-Alignment: <x> $\neq$ /x/

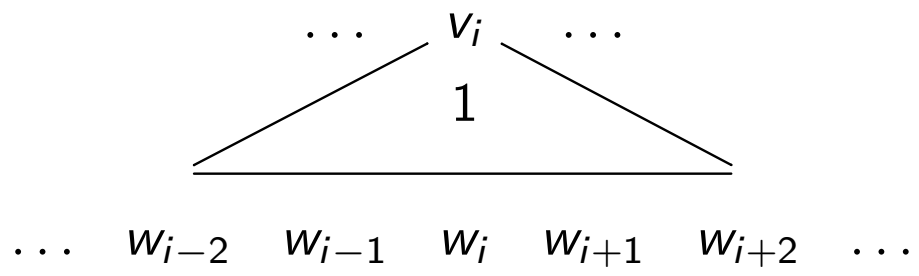
Alinierung: Kostenfunktion

Statistisch

- Ermittlung der **Editierkosten** $c(x, y)$ über Graphem-Phonem-Kookkurenzen.
  Für Graphem v_i, Phonem w_j:
  - **Substitution:** $c(v_i, w_j) = 1 - P(w_j|v_i)$
  - **Löschung:** $c(v_i, \_) = 1 - P(\_|v_i)$
  - **Einfügung:** $c(\_, w_j) = 1 - P(w_j|\_)$
  - **MLE**: $P(A|B) = \frac{\#(A,B)}{\#(B)}$

Alinierung: Kostenfunktion

- Ermittlung der **Häufigkeiten** $\#(*)$
  - Verteilung des Inkrements 1 in auf v_i zentriertem Dreiecksfenster



  - Auffüllen der kürzeren Sequenz mit __-Zeichen und __-Permutation

s	c	h	u	l	e
__	__	S	u:	l	@
__	S	__	u:	l	@
S	__	__	u:	l	@
		. . .			

Alinierung: Kostenfunktion

- Normalisierung des Δ-gewichteten Inkrements auf Anzahl der Permutationen.

$\longrightarrow$ **einheitliche Behandlung von Substitutionen, Einfügungen und Auslassungen**

  - Einfügung: _–w_j-Substitution
  - Auslassung: v_i–_-Substitution

- Anzahl der Permutationen: k mal _, in einer Sequenz der Länge n: $\binom{n}{k} = \frac{n!}{k!\cdot(n-k)!}$. Im obigen Beispiel: $\binom{6}{2} = \frac{6!}{2!\cdot 4!} = 15$

Alinierung: Kostenfunktion

Heuristiken

- Permutationsconstraint: nicht mehr als zwei aufeinanderfolgende _-Zeichen
- maximale Längendifferenz: $\mathrm{abs}(|v| - |w|) \leq d$

Konvertierung: Table Lookup with Defaults

Table Lookup with Defaults (van den Bosch et al., 1993)

- **Training:** Speicherung des jeweils kürzesten Graphemkontexts für ein eindeutiges Graphem-Phonem-Mapping in Tabelle G

k **a** m m	/a/
u ß	/u:/

- **Sortierung** nach Länge der Graphemsequenz
- **2 Default-Tabellen:** Graphem-Fenster + am häufigsten damit ko-okkurierendes Phonem

$v_{i-1} v_i v_{i+1}$	w_j
v_i	w_j

Konvertierung: Table Lookup with Defaults

- **Konvertierung:**
  - Suche nach passendem Graphem-Muster (von lang nach kurz) in Tabelle G
  - Falls nicht vorhanden, Rückgriff auf Default-Tabellen
  - Beispiel:
    - zu konvertieren: $<u>$ in *Fuß*
    - in Tabelle G gefundenes Muster: *uß*

    $\longrightarrow$ Ausgabe: */u:/*
- **Vorzüge**
  - rein datenbasierter Ansatz

  $\longrightarrow$ kein Expertenwissen nötig, sprachunabhängig

Maschinelles Lernen

- **Ziel:** Erlernen des Zusammenhangs zwischen Zielwerten (Kategorien oder kontinuierliche Werte) für Objekte und deren Eigenschaften.
- bezogen auf Graphem-Phonem-Konvertierung
  - **Objekte:** Grapheme
  - **Eigenschaften:** Graphem-Identität, umgebende Grapheme, Position des Graphems innnerhalb der Silbe, ...
  - **Zielwerte:** Phoneme

Maschinelles Lernen

- Objekte als **kategorisierte Merkmalsvektoren** (*Featurevektoren*) repräsentiert.
  - **unabhängige Variablen (Attribute)** für Graphem v_i: $[v_{i-1}, v_i, v_{i+1}$, Morphemtyp, $\_\#_\sigma]$
  - Attributwerte: [a–z, a–z, a–z, *frei*|*gebunden*, 0|1]
  - Merkmalsvektor für erstes <*e*> in *g***e***ben*: [*g, e, b, frei, 1*]
  - **Kategorie (abhängige Variable):** */e:/*

Maschinelles Lernen

- **Variablenwerte:** kategorial oder kontinuierlich
  - **kategorial:** Graphem-Identität, Position in Silbe, Phonemklasse, Wortbetonung
  - **kontinuierlich:** relative Position des Graphems im Wort, Lautdauer, F0-Wert

Maschinelles Lernen

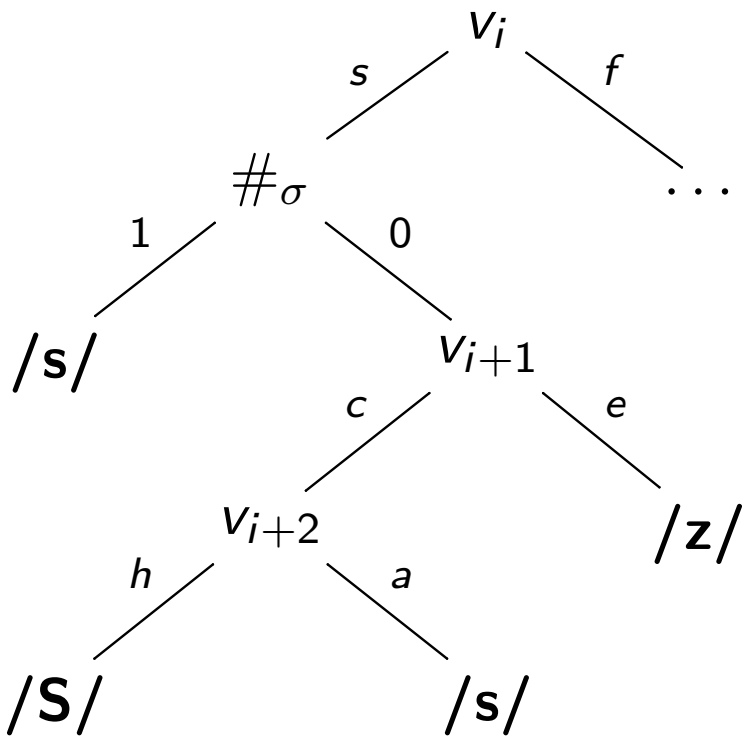
- **Überwachtes Lernen:** Werte der abhängigen Variable in Trainingsdaten bekannt; C4.5, CART, neuronale Netze (ANN)
- **Unüberwachtes Lernen:** Werte nicht bekannt; Clustering, ANN
- **Variablentypen:**
  - C4.5: kategorial/kontinuierlich $\longrightarrow$ kategorial; z.B. Akzent
  - CART: kategorial/kontinuierlich $\longrightarrow$ kategorial/kontinuierlich; z.B. Lautdauer
  - ANN: kontinuierlich $\longrightarrow$ kategorial/kontinuierlich

Entscheidungsbäume

- Quinlan (1993); http://www.cse.unsw.edu.au/~quinlan
- **Modellierung:**
  - **Objekt:** Pfad durch den Baum
  - **Eigenschaften:** nonterminale Knoten (Attribute) + Kanten (Werte)
  - **Zielwerte:** terminale Knoten
- Vorteil: Transparenz $\longrightarrow$ Wissensakquirierung möglich

Entscheidungsbäume

Beispielausschnitt:



Entscheidungsbäume

Rekursiver Aufbau des Baums

- **Fall 1:** Gehören alle Objekte, die noch nicht durch einen vollständigen Pfad im Baum repräsentiert sind, der **gleichen Klasse** an, so erzeuge ein Blatt und ordne die Objekte diesem Blatt zu.
- **Fall 2:** Verfahre genauso, wenn die Objekte **verschiedenen Klassen** angehören, sich **aber** anhand der gegebenen Attribute **nicht mehr weiter unterscheiden** lassen.
- **Fall 3:** Gehören die Objekte **verschiedenen Klassen** an **und unterscheiden** sie sich in einer oder mehreren Eigenschaften, so wähle das **zur Partitionierung der Objektmenge am 'besten geeignete' Attribut** und erzeuge einen Knoten, an dem sich der Baum in mehrere durch Werte des betrachteten Attributs vorgegebene Kanten aufspaltet.

Entscheidungsbäume

Rekursion:

- Wiederholte Anwendung einer Handlung auf ihr eigenes Ergebnis.
- Handlung hier: Erzeugung neuer Knoten und davon abgehender Kanten (**Fall 3**)
- Realisierung:
  - sich selbst aufrufende Funktion
  - Abbruchkriterien (**Fall1, Fall2**) $\longrightarrow$ abschließende Erzeugung von **terminalen Knoten (Blättern)**

Entscheidungsbäume

- **Attribute:** vorangehendes und aktuelles Graphem X_{i-1}, X_i
- **Kategorie:** Phonem Y
- **Fall 1:**

X_{i-1}	X_i	Y
e	s	s
o	s	s

$\longrightarrow$ **Zuordnung der Objekte zum selben /s/-Blatt.**

- **Fall 2:**

X_{i-1}	X_i	Y
e	s	s
e	s	s
e	s	z

$\longrightarrow$ **Zuordnung der Objekte zum selben /s/-Blatt (häufigste Kategorie).**

Entscheidungsbäume

Fall 3: Bestimmung des besten Attributs

- Das beste Attribut liefert den höchsten **Informationsgewinn**.
- **Entropie:** durchschnittlicher **Informationsgehalt** einer Variablen

$$H(C) = -\sum_{c \in C} p(c) \log_2 p(c) \quad [Bit] \tag{4}$$

C: Menge aller Objektklassen, $p(c)$: Wahrscheinlichkeit der Klasse $c \in C$. (C=Variable, c=Variablenbelegung).
Angabe, wieviel Information im Durchschnitt benötigt wird, um die Klasse eines Objekts vorhersagen zu können

Entscheidungsbäume

- **Bedingte Entropie**

$$
\begin{aligned}
H(C|A) &= \sum_{a \in A} p(a) H(C|A=a) \\
&= \sum_{a \in A} p(a) \Big[- \sum_{c \in C} p(c|a) \log_2 p(c|a) \Big] \qquad (5)
\end{aligned}
$$

Angabe, wieviel Information im Durchschnitt **zusätzlich zu dem Wissen darüber, daß das Attribut A den Wert a hat**, nötig ist, um die Klasse $c \in C$ eines Objekts vorhersagen zu können.

Entscheidungsbäume

Veranschaulichendes Beispiel:

- **vorherzusagen:** $C = \{$*'Straße nass'*, *'Straße trocken'*$\}$
- Attribut $A_1 = \{$*'es regnet'*, *'es ist sonnig'*$\}$
- Attribut $A_2 = \{$*'Ampel rot/gelb'*, *'Ampel grün'*$\}$
- **Wahrscheinlichkeiten:**

$$P(\text{nass}|\text{Regen}) = 1 \quad P(\text{nass}|\text{Sonne}) = 0 \quad P(\text{trocken}|\text{Regen}) = 0 \quad P(\text{trocken}|\text{Sonne}) = 1$$

$$P(\text{nass}|\text{rot}) = 0.5 \quad P(\text{nass}|\text{grün}) = 0.5 \quad P(\text{trocken}|\text{rot}) = 0.5 \quad P(\text{trocken}|\text{grün}) = 0.5$$

$$P(\text{Regen}) = 0.5 \quad P(\text{Sonne}) = 0.5 \quad P(\text{rot}) = 0.5 \quad P(\text{grün}) = 0.5$$

Entscheidungsbäume

- **Frage: Welche Variable A_1 oder A_2 beinhaltet mehr Information über Variable C?**
- **Intuitive Antwort: A_1, da**
  1. sich anhand der gegebenen Wahrscheinlichkeiten sicher vorhersagen lässt, ob die Straße nass oder trocken ist, wenn bekannt ist, ob es regnet oder nicht, und
  2. das Wissen über das Ampelsignal nichts zur Vorhersage des Straßenzustands beiträgt (alle Wahrscheinlichkeiten gleich 0.5).
- **Rechnerische Antwort:** A_1, da das Einsetzen der Wahrscheinlichkeiten in Gleichung (5) Folgendes ergibt:
  - $H(C|A_1) = 0$
  - $H(C|A_2) = H(C)$

Entscheidungsbäume

- nachrechnen als Übung für daheim, Anmerkung: $0 \cdot \log_2 0 := 0$
- **Interpretation**
  - $H(C|A_1) = 0$: wenn das *Wetter* A_1 bekannt ist, wird keine zusätzliche Information mehr zur Vorhersage der *Straßennässe* C benötigt.
  - $H(C|A_2) = H(C)$: in dem Wissen über das *Ampelsignal* A_2 steckt überhaupt keine Information über die *Straßennässe* C.
- **Allgemeine Folgerung:** je stärker die bedingten Wahrscheinlichkeiten $P(c|a)$ von einer Gleichverteilung abweichen, desto mehr Information über C steckt in A.

Entscheidungsbäume

- **Informationsgewinn:** umso größer, je mehr Information über C in A steckt

$$G(A) = H(C) - H(C|A) \tag{6}$$

- **gewählt wird also Attribut** $\widehat{A} = argmax_A[G(A)]$
- Attribute mit *kontinuierlichen* Werten werden anhand von Trennwerte t kategorisiert in Klassen "$> t$" und "$\leq t$"

Entscheidungsbäume

Pruning

- **Beschneidung** des Baums vs. **Überadaption** an Trainingsdaten
- z.B. mittles Evaluierung anhand eines **Entwicklungskorpus**
- Verringern der Verästelungstiefe solange, bis sich Performanz des Baums auf Entwicklungskorpus verschlechtert

Entscheidungsbäume

Beispiel zur Wahl des besten Attributs

- **Attribute:**
  - aktuelles Graphem $X_i = \{s,g\}$
  - vorangehendes Graphem $X_{i-1} = \{e,o\}$
- **Kategorie:** Phonem $Y = \{s,g,k\}$
- **Objekte:**

Merkmalsvektor		Kategorie
X_{i-1}	X_i	Y
e	s	s
o	s	s
o	g	g
e	g	g
e	g	k

Entscheidungsbäume

- **Wahrscheinlichkeiten anhand der Objekt-Tabelle:**

$p(\mathrm{y})$	$p(\mathrm{x}_i)$	$p(\mathrm{x}_{i-1})$	$p(\mathrm{y}\|\mathrm{x}_i)$	$p(\mathrm{y}\|\mathrm{x}_{i-1})$
$p(\mathrm{s}) = 0.4$	$p(\mathrm{s}) = 0.4$	$p(\mathrm{e}) = 0.6$	$p(\mathrm{s}\|\mathrm{s}) = 1$	$p(\mathrm{s}\|\mathrm{e}) = 0.\overline{3}$
$p(\mathrm{g}) = 0.4$	$p(\mathrm{g}) = 0.6$	$p(\mathrm{o}) = 0.4$	$p(\mathrm{s}\|\mathrm{g}) = 0$	$p(\mathrm{s}\|\mathrm{o}) = 0.5$
$p(\mathrm{k}) = 0.2$			$p(\mathrm{g}\|\mathrm{s}) = 0$	$p(\mathrm{g}\|\mathrm{e}) = 0.\overline{3}$
			$p(\mathrm{g}\|\mathrm{g}) = 0.\overline{6}$	$p(\mathrm{g}\|\mathrm{o}) = 0.5$
			$p(\mathrm{k}\|\mathrm{s}) = 0$	$p(\mathrm{k}\|\mathrm{e}) = 0.\overline{3}$
			$p(\mathrm{k}\|\mathrm{g}) = 0.\overline{3}$	$p(\mathrm{k}\|\mathrm{o}) = 0$

Entscheidungsbäume

- **Entropie von Y**

$$
\begin{aligned}
H(\mathrm{Y}) &= -\sum_{\mathrm{y}\in\mathrm{Y}} p(\mathrm{y}) \log_2 p(\mathrm{y}) \\
&= -\Big[p(\mathrm{s}) \log_2 p(\mathrm{s}) + p(\mathrm{g}) \log_2 p(\mathrm{g}) + p(\mathrm{k}) \log_2 p(\mathrm{k})\Big] \\
&= -\Big[0.4 \log_2 0.4 + 0.4 \log_2 0.4 + 0.2 \log_2 0.2\Big] \\
&= 1.5219
\end{aligned}
$$

- d.h. es werden zur Codierung eines $\mathrm{Phonems}$ aus Y durchschnittlich 1.5219 Bit benötigt.

Entscheidungsbäume

- **Bedingte Entropie von Y gegeben Attribut X_i**

$$
\begin{aligned}
H(\mathrm{Y}|\mathrm{X}_i) &= \sum_{\mathrm{x}\in\mathrm{X}_i} p(\mathrm{x})\Big[-\sum_{\mathrm{y}\in\mathrm{Y}} p(\mathrm{y}|\mathrm{x})\log_2 p(\mathrm{y}|\mathrm{x})\Big] \\
&= p(\mathrm{s})\Big[-\Big(p(\mathrm{s}|\mathrm{s})\log_2 p(\mathrm{s}|\mathrm{s}) + p(\mathrm{g}|\mathrm{s})\log_2 p(\mathrm{g}|\mathrm{s}) + p(\mathrm{k}|\mathrm{s})\log_2 p(\mathrm{k}|\mathrm{s})\Big)\Big] + \\
&\quad p(\mathrm{g})\Big[-\Big(p(\mathrm{s}|\mathrm{g})\log_2 p(\mathrm{s}|\mathrm{g}) + p(\mathrm{g}|\mathrm{g})\log_2 p(\mathrm{g}|\mathrm{g}) + p(\mathrm{k}|\mathrm{g})\log_2 p(\mathrm{k}|\mathrm{g})\Big)\Big] \\
&= 0.4\cdot\Big[-\Big(1\cdot\log_2 1 + 0 + 0\Big)\Big] + 0.6\cdot\Big[-\Big(0 + 0.\bar{6}\cdot\log_2 0.\bar{6} + 0.\bar{3}\cdot\log_2 0.\bar{3}\Big)\Big] \\
&= \mathbf{0.5510}
\end{aligned}
$$

- d.h. es werden zusätzlich zur Kenntnis des aktuellen Graphems **0.5510** Bit zur Vorhersage des Phonems benötigt
- **Anmerkung:** $0 \cdot log_2 0 := 0$ (oder Smoothing)

Entscheidungsbäume

- **Bedingte Entropie von Y gegeben Attribut X_{i-1}**

$$
\begin{aligned}
H(\mathrm{Y}|\mathrm{X}_{i-1}) &= p(\mathrm{e})\Big[-\Big(p(\mathrm{s}|\mathrm{e})\log_2 p(\mathrm{s}|\mathrm{e}) + p(\mathrm{g}|\mathrm{e})\log_2 p(\mathrm{g}|\mathrm{e}) + p(\mathrm{k}|\mathrm{e})\log_2 p(\mathrm{k}|\mathrm{e})\Big)\Big] + \\
&\quad p(\mathrm{o})\Big[-\Big(p(\mathrm{s}|\mathrm{o})\log_2 p(\mathrm{s}|\mathrm{o}) + p(\mathrm{g}|\mathrm{o})\log_2 p(\mathrm{g}|\mathrm{o}) + p(\mathrm{k}|\mathrm{o})\log_2 p(\mathrm{k}|\mathrm{o})\Big)\Big] \\
&= 0.6 \cdot \Big[-\Big(0.\bar{3}\cdot\log_2 0.\bar{3} + 0.\bar{3}\cdot\log_2 0.\bar{3} + 0.\bar{3}\cdot\log_2 0.\bar{3}\Big)\Big] + \\
&\quad 0.4 \cdot \Big[-\Big(0.5\cdot\log_2 0.5 + 0.5\cdot\log_2 0.5 + 0\Big)\Big] \\
&= \mathbf{1.3510}
\end{aligned}
$$

- d.h. es werden zusätzlich zur Kenntnis des vorangehenden Graphems **1.3510** Bit zur Vorhersage des Phonems benötigt

Entscheidungsbäume

- **Attributabhängiger Informationsgewinn** G:

$$G(\mathrm{X}_i) = H(\mathrm{Y}) - H(\mathrm{Y}|\mathrm{X}_i) = 1.5219 - 0.5510 = 0.9709$$
$$G(\mathrm{X}_{i-1}) = H(\mathrm{Y}) - H(\mathrm{Y}|\mathrm{X}_{i-1}) = 1.5219 - 1.3510 = 0.1709$$

$\longrightarrow$ mit aktuellem Graphem verbundener Informationsgewinn über das Phonem ist höher als der des vorangehenden Graphems

$\longrightarrow$ verwende X_i zur Aufteilung der Objekte

Entscheidungsbäume

G2P-Anwendung von Entscheidungsbäumen

- **Features:**
  - Graphemkontext
  - Silben-Features: Aufbau der Silbe (nacht/bedeckt, offen/geschlossen), Position in Silbe (Onset, Nukleus, Coda, Gelenk)
  - morphologische Features: Morphemklasse, +/– folgende Morphemgrenze
  - Phonem-Vorgeschichte
- **Vorhersage**: Phonem, incl. leeres Phonem (/_/), Phonem-Cluster (/?+a:/)

Morphologische Zerlegung

1. Teile jedes Wort *w* rekursiv von links nach rechts in String-Präfixe und -Suffixe bis eine erlaubte Segmentierung möglich ist oder das Wortende erreicht wird.
2. Im Laufe der Rekursion wird eine Grenze, die den aktuellen String in Präfix und Suffix unterteilt dann akzeptiert wenn (i) das Präfix im Lexikon zu finden ist, (ii) eine erlaubte Segmentierung des Suffixes möglich ist, oder falls nicht das Suffix im Lexikon steht, (iii) die Sequenz 'Präfix-Klasse + Klasse des ersten Suffixes' nicht der Morphotaktik widerspricht und (iv) die Klasse des letzten Suffixes kompatibel ist mit dem POS von *w*.

Morphologische Zerlegung

Beispiel: *Fassade – nkletterer*

(i) String-Präfix *Fassade* im Lexikon

(ii) **Rekursion:** erlaubte Segmentierung des String-Suffixes *nkletterer* möglich (*n-kletter-er*)

(iii) Morphemklassen *Fassade/NN – n/Fugenmorphem* kompatibel

(iv) Morphemklasse des letzten String-Suffixes *er/NN-Suffix* kompatibel mit POS NN des Worts

$\longrightarrow$ Segmentierung *Fassade – nkletterer* möglich

Silbifizierung

In Graphemfolge

- **3 vorherzusagende Klassen:**
  Silbengrenze folgt, folgt nicht, Ambisyllabizität
- **Feature-Auswahl:**
  - für jedes Graphem v_i
  - innerhalb eines auf v_i zentrierten symmetrischen Graphem-Fensters
  - Graphem, Konsonant/ Vokal
  - ggf. Morphemgrenze (+/− relevant für Silbengrenze)

Silbifizierung

- für Silbifizierung **relevante morphologische Grenzen**:
  - vor allen Morphemen außer Flexionsendungen, Suffixen, Komparationsmorphemen und Fugen
  - vor Flexionsendungen, Suffixen mit initialem Konsonanten und eigenem Silbenkern (*schaff***te**)
  - vor Flexionsendungen, Suffixen mit initialem Vokal, wenn das vorangehende Morphem auf Vokal endet (*bau***en**)

Silbifizierung

In Phonemfolge

1. setze vor jedes Sonoritätsminimum eine Silbengrenze
2. Feinadjustierung gemäß Kohlers (1995) Silbenphonotaktik und silbengrenzrelevanter Morphemgrenzen

Beispiel:

/fE6hEltnIs/ $\xrightarrow{1.}$ /fE6.hEl.tnIs/ $\xrightarrow{2.}$ /fE6.hElt.nIs/

Silbifizierung

Silbenphonotaktik (Kohler, 1995)

$$\left(\left\{ \begin{array}{ccc} & K_{a,b,c} & \\ (K_a) & K_a & K_b \\ & K_a & K_c \\ (K_a) & K_a & K_a \end{array} \right\} \; V \; \left\{ \begin{array}{ccc} & K_{a,b} & \\ K_b & K_a & (K_a) \\ K_b & K_b & (K_a) \\ K_a & K_a & \end{array} \right\} \right) \left(\left\{ \begin{array}{c} K_a(+K_a) \\ +K_a(K_a) \end{array} \right\} \right)$$

- K_a: Plosive, Frikative
- K_b: Nasale, /l/, /r/
- K_c: /h/, /j/
- V: Vokale
- $+$: Morphemgrenze

Silbifizierung

Silbenbeispiele:

- *Herbsts* /hE6psts/ $K_c V K_a K_a + K_a$
- *Psalm* /psalm/ $K_a K_a V K_b K_b$

Restriktionen gegen Übergeneralisierung

- **Beispiel:** für $(K_a)K_a K_b$ darf K_a nicht im Artikulationsort mit K_b übereinstimmen,
  vgl. /fE6.hEl.tnls/ $\longrightarrow$ /fE6.hElt.nls/

Wortbetonung

Simplex-Wörter: Restriktionen

- **Drei-Silben-Fenster**: Wortbetonung kann nur auf eine der letzten drei Silben im Wort fallen (Ultima, Penultima, Antepenultima)
  Ausnahme: *schwa* in Antepenultima (/ 'a:.b@n.tOY.6 /)
- **Closed penult**: geschlossene Penultima verhindern eine Betonung weiter links (/ hi.b'Is.kUs /)
- **Final schwa**: Betonung der Penultima, wenn der Silbenkern der Ultima aus einem Schwa besteht: / tsi.tr'o:.n@ /;
  Produktivität: / g'e:.nE.zIs / $\longrightarrow$ / gE.n'e:.z@ /

Wortbetonung

Allgemeine Tendenzen im Standard-Deutschen

- Wortbetonung eher hinten im Wort
- eher auf *schweren* Silben (Langvokal/ Diphtong, Coda)
- *schwa*-Silben nicht betonbar

Wortbetonung

Simplex+Affixe

- **betonte Morphemklassen:**
  - Verbpartikeln: **w'eg***fahren*
  - betonte Affixe: *Abstin'***enz**, *pass'***abel**
- **unbetonte Morphemklassen:**
  - Flektionsendungen
  - unbetonte Affixe: **ent***spr'echen*, *M'ann***schaft**, *'Ärge***mis**

Komposita

- häufig Haupt- und Nebenbetonung
- **zweigliedrig:** Hauptbetonung auf erstem Glied (wenige Ausnahmen: *Lebew'ohl*)

Wortbetonung

- **mehrgliedrig:** Anwendung der **Compound-stress-Rule (CSR)** der metrischen Phonologie:

  CSR: *im Kompositum AB ist B strong s, wenn es sich weiter verzweigt, ansonsten ist A strong und B weak w*

- Erklärung von **Ausnahmen:** Atomisierung lexikalisierter verzweigender Konstitutenten: *K'unst#[hand#werk]*, daher: drei– $\longrightarrow$ zweigliedrig $\longrightarrow$ Hauptbetonung auf erstem Glied

Wortbetonung

weitere Schwierigkeiten

- **stress shift**: *D'oktor* > *Dokt'oren* (vgl. *Final schwa*-Restriktion); *'Ablauf* > *Progr'amm#abl"auf*
- **Homographen** unterschiedlicher Wortart, Valenz
  - *mod'ern/ADJ* **vs.** *m'odern/V*
  - *K'onstanz/NE* **vs.** *Konst'anz/NN*
  - *'allerhand/Indefpron,ADV* **vs.** *allerh'and/ADJD*
  - *d'arüber/PAV* **vs.** *dar'über/PTKVZ*
  - *d'urchlaufen/V(intrans)* **vs.** *durchl'aufen/V(trans)*

Wortbetonung

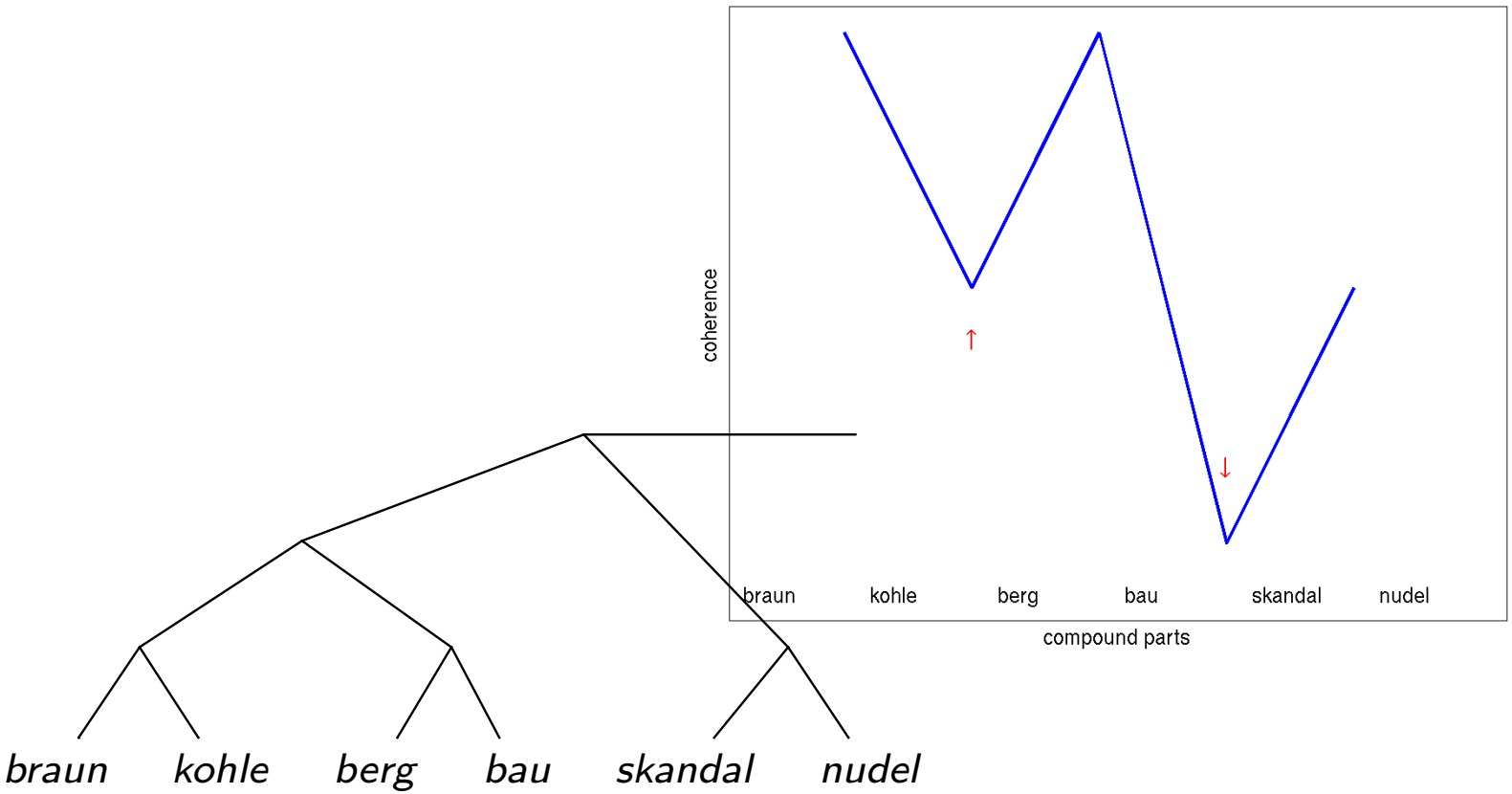
- **Kontrastakzent:**
  - *'Arbeitgeber* **vs.** *Arbeitg'eber und Arbeitn'ehmer*
  - *R'egel* **vs.** *ich habe Reg'el gesagt, nicht Regal*
- **unklare Fälle:** *m'erkwürdigerweise* **vs.** *merkwürdigerw'eise*

Wortbetonung

Vorhersage

1. Kompositumzerlegung (mittels morphologischer Analyse, s.o.)
2. Bestimmung des betonten Kompositumglieds (mittels metrischer Bäume)
3. Lokalisierung der betonten Silbe im betonten Kompositumglied (mittels morphologischer Analyse zur Identifizierung betonter Affixe und maschinellem Lernen)

Betontes Kompositumglied: Induktion metrischer Bäume



Kohärenz: z.B. Bigramm-Wahrscheinlichkeiten: $P(\text{skandal}|\text{bau}) < P(\text{bau}|\text{berg})$

Wortbetonung: Lokalisierung der betonten Silbe

Mittels maschinellem Lernen

- **Objekte:** Silben oder Wörter (Kompositumglieder)
- **Zielwerte:**
  - für Silben: $+/-$ betont
  - für Wörter: Index der betonten Silbe

Wortbetonung: Lokalisierung der betonten Silbe

Instanzbasiertes Lernen (Mustervergleich; Daelemans et al., 1994)

- **Training:** Abspeichern von Wörtern in Form von Merkmalsvektoren zusammen mit Index der betonten Silbe
- **Merkmale:** z.B. Silbengewicht "**s**chwer, **l**eicht" der letzten beiden Silben; **Zielwerte: u**ltima, **p**enultima, **a**ntepenultima

Wort	Merkmalsvektor	Betonung
Wanne	s l	p
Sonne	s l	p
Hibiskus	s s	p
genau	l s	u

Wortbetonung: Lokalisierung der betonten Silbe

- **Anwendung** – für Wort *w*: Übernahme des häufigsten Betonungsmusters unter den *w*-ähnlichsten Wörtern
- **Hamming-Distanz d** zweier Merkmalsvektoren: Anzahl der unterschiedlichen Werte

  w = Wonne (Merkmalsvektor: [s l])
  d(Wonne,Wanne)=0, d(Wonne,Sonne)=0,
  d(Wonne,Hibiskus)=1, d(Wonne,genau)=2
  $\longrightarrow$ Übernahme der Betonung der ähnlichsten Wörter *Wanne*, *Sonne*, also *penultima*

Wortbetonung: Lokalisierung der betonten Silbe

Entscheidungsbaum

- Objekte: Silben
- Features: Silbengewicht, Morphemklasse, POS, Silbenindex, Position in Kompositum
- Zielwert: +/– betont