Words and Wordforms

- Lexical items
- Dictionary lookup
- Word segmentation
- Morphological analysis
- Morphophonology
- Lexical semantics
- Distributed representations
- Part-of-speech tagging
- Word-sense disambiguation

Words and Wordforms

- Lexical items
- Dictionary lookup
- Word segmentation
- Morphological analysis
- Morphophonology
- Lexical semantics
- Distributed representations
- Part-of-speech tagging
- Word-sense disambiguation

Lexical items

- wordform: string of characters delimited by blanks (or interpunction symbols)
- lexeme: set of wordforms corresponding to the same dictionary entry
- lemma: canonical form describing the lexeme

- wordforms are created from morphemes
- morphemes carry (syntactic/semantic/pragmatic) information
- they differentiate the meaning of words and wordforms

Lexical items

Lexical processes

- free/unbound morphemes: may form a word alone, possibly inflected
  - usually corresponds to the root of a word:

    car, *Tisch*, ...

- bound morphemes: can only be used together with an unbound one:

  im-, *-ly*, *un-*, *-ung*, ...

- new lexemes can be produced by derivation and compounding
- new wordforms can be produced by inflection

Lexical items

Inflection

- creating different wordforms for the same lexeme

  sleep, *sleeps*, *slept*, *sleeping*

  schlafe, *schläfst*, *schläft*, *schlafen*, *schlaft*,
  schlief, *schliefst*, *schlief*, *schliefen*, *schlieft*

- the set of wordforms is also called a paradigm
- inflection never affects the grammatical category ...
- ... but might determine but some morpho-syntactic features (e.g. case, number, tense, ...)

Lexical items

Derivation

- creating new lexemes from existing ones by adding affixes, possibly zero-affixes

  sleep, sleeper, asleep, sleepy, sleepyness, sleepless, ...

  Schlaf, Schläfer, schläfrig, Schläfrigkeit,
  verschlafen, ausschlafen, durchschlafen, einschlafen,
  geschlafen, ausgeschlafen, auszuschlafen,
  schlaflos, Schlaflosigkeit, ...

Lexical items

- derivation may change the grammatical category and affects syntactic, semantic and pragmatic properties of a word
  - e.g. negation (adjective $\rightarrow$ adjective)

    the happy man
    the unhappy man

    der glückliche Mann
    der unglückliche Mann
  - e.g. nominalization (verb $\rightarrow$ noun)

    we will investigate these issues
    we will carry out an investigation of these issues

    wir werden diese Fragen untersuchen
    wir werden eine Untersuchung dieser Fragen durchführen

Lexical items

Compounding

- combining two lexemes to form a new one

  Schlafzimmer, *Schlafanzug*, *Schlaftablette*, *Schlafstörung*, *Schlafsofa*, *Tiefschlaf*, *Vorlesungsschlaf*, ...

  bedroom, *nightgown*, *sleep disturbances*, *sofa bed*, ...

- the compound takes the syntactic properties of its rightmost component
- the semantic relationship between the components is highly ambiguous *Sonnenschutz*, *Jugendschutz*, *Arbeitsschutz*,
- compounds can always be paraphrased

  Schlafzimmer $\rightarrow$ *Zimmer zum Schlafen*

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  $bank \rightarrow my\ bank\ account$

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank $\rightarrow$ *my bank account*

  Frau

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank $\rightarrow$ *my bank account*

  Frau $\rightarrow$ *die Frau*

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank $\rightarrow$ *my bank account*

  Frau $\rightarrow$ *die Frau* $\rightarrow$ *die Frau steht*

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank $\rightarrow$ *my bank account*

  Frau $\rightarrow$ *die Frau* $\rightarrow$ *die Frau steht*

  Brücke

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank $\rightarrow$ *my bank account*

  Frau $\rightarrow$ *die Frau* $\rightarrow$ *die Frau steht*

  Brücke $\rightarrow$ *auf der Brücke*

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank $\rightarrow$ *my bank account*

  Frau $\rightarrow$ *die Frau* $\rightarrow$ *die Frau steht*

  Brücke $\rightarrow$ *auf der Brücke* $\rightarrow$ *auf der Brücke über die Bahnlinie*

Lexical items

- lexical information is highly ambiguous if wordforms are considered in isolation
- contextual information can reduce or remove the ambiguity

  bank $\rightarrow$ *my bank account*

  Frau $\rightarrow$ *die Frau* $\rightarrow$ *die Frau steht*

  Brücke $\rightarrow$ *auf der Brücke* $\rightarrow$ *auf der Brücke über die Bahnlinie*

- computational approaches
  - classifiers for symbol sequences (e.g. hidden MARKOV models)
  - information accumulation in unification-based grammars

Words and Wordforms

- Lexical items
- Dictionary lookup
- Word segmentation
- Morphological analysis
- Morphophonology
- Lexical semantics
- Distributed representations
- Part-of-speech tagging
- Word-sense disambiguation

Words and Wordforms

- Lexical items
- Dictionary lookup
- Word segmentation
- Morphological analysis
- Morphophonology
- Lexical semantics
- Distributed representations
- Part-of-speech tagging
- Word-sense disambiguation

Dictionary lookup

- Tries
- Tries as FSAs
- Probabilistic Tries
- String similarity

Dictionary lookup

- use cases
  - fetching the information associated with the keyword
    - translation equivalents, grammatical features, frequency, position of occurrence in a text, prounnciations, ...
  - confirming the identity of the keyword
    - spell checking, wordform completion, ...

- simplest approach: string identity
- makes strong assumptions about the correctness of the keyword
  - unrealistic for many real world scenarios
- relaxed requirements: word (form) similarity
  - prefix matching
  - elastic string matching

Dictionary lookup

- prefix matching for wordform completion (Kushler et al. 1998)
- beneficial for incremental, but underspecified or uncertain data input
  - e.g. T9 keyboard (Kushler et al. 1998)
  - tree-structured pronunciation dictionaries for speech recognition
- useful data structure: Trie (Fredkin 1960)

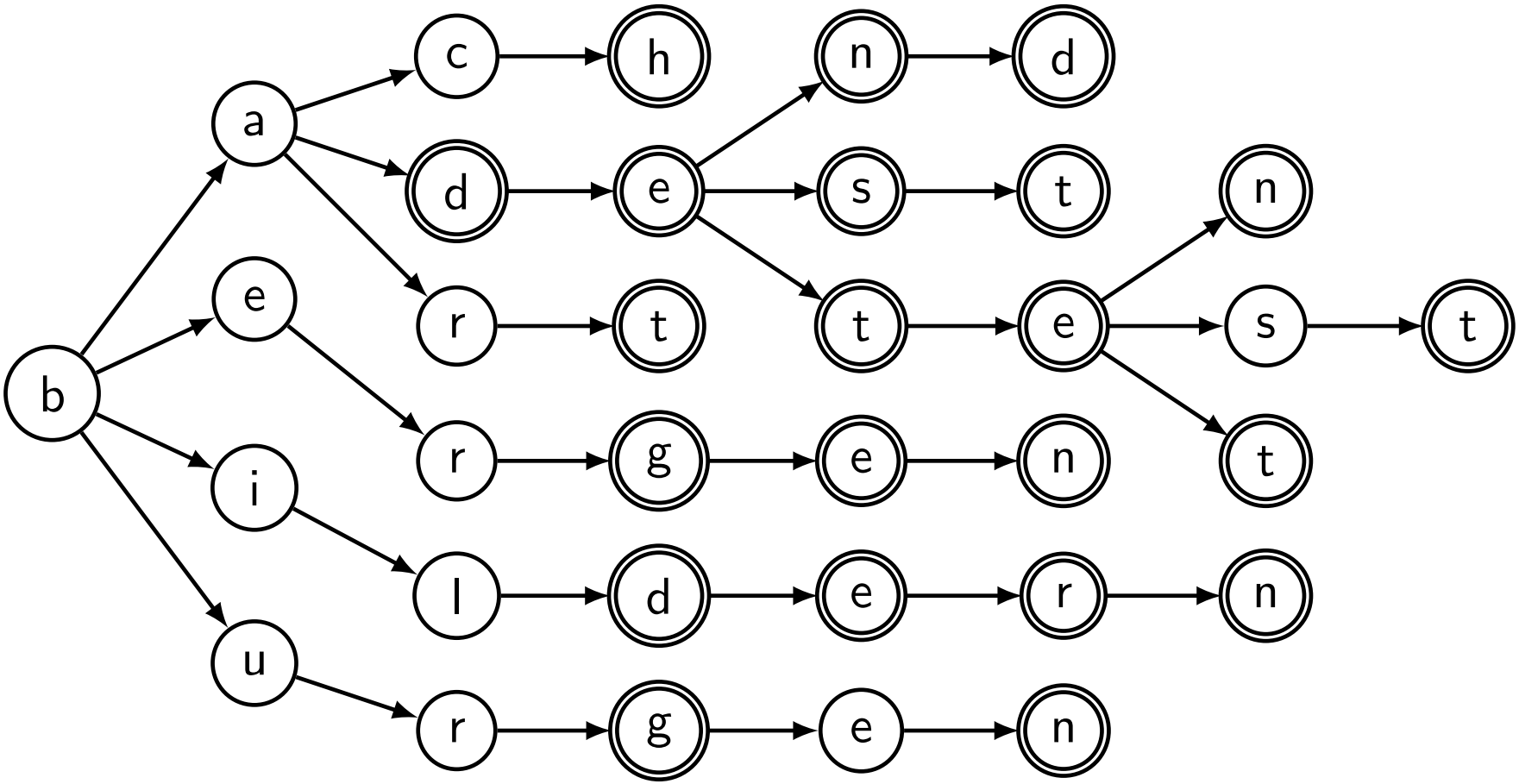
Tries

- tree structure with
  - nodes corresponding to single characters
  - edges encoding possible continuations of a string prefix
  - marked nodes denoting complete keywords

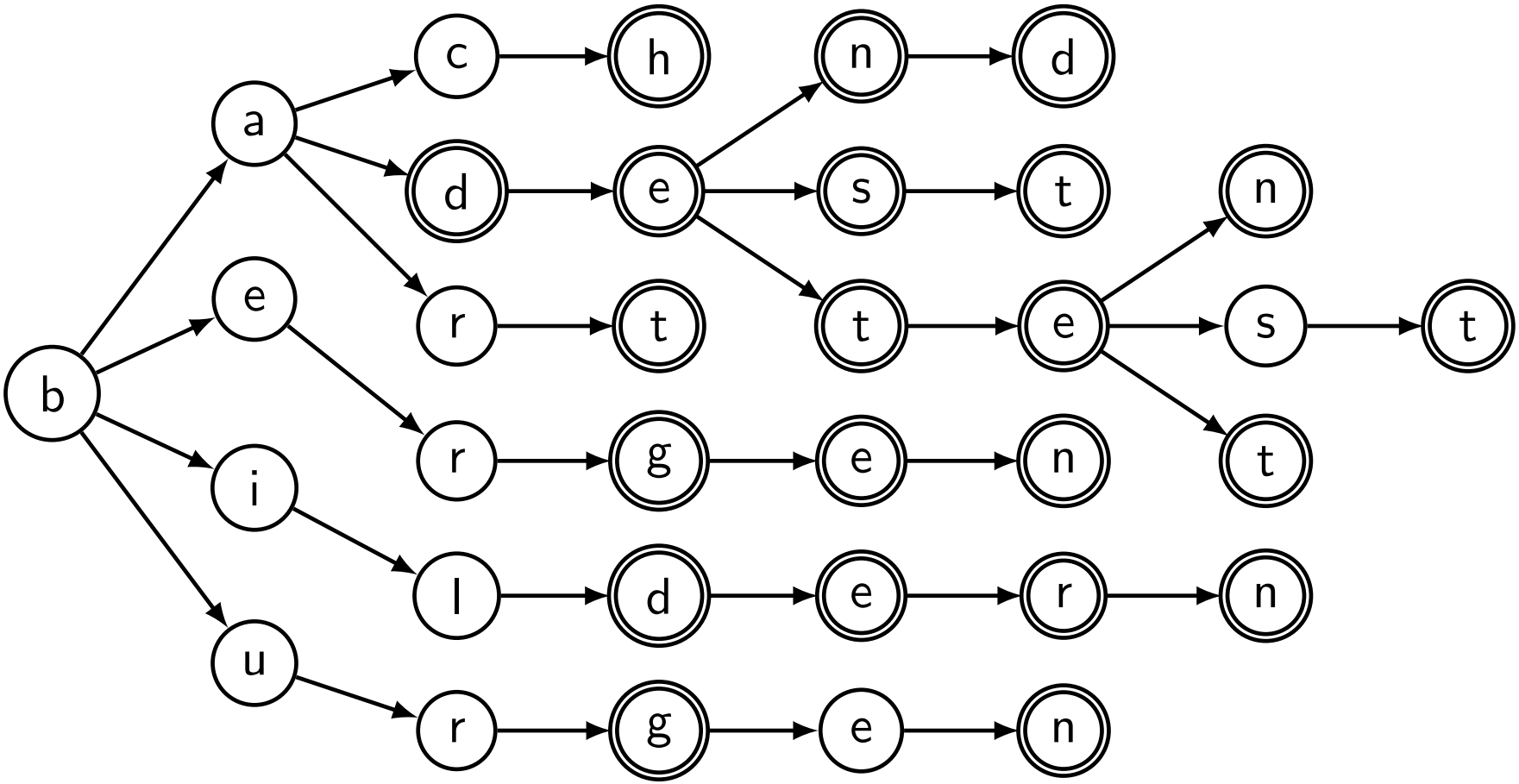
Tries

- tree structure with
  - nodes corresponding to single characters
  - edges encoding possible continuations of a string prefix
  - marked nodes denoting complete keywords
- corresponds to a finite state automaton

Tries as FSAs

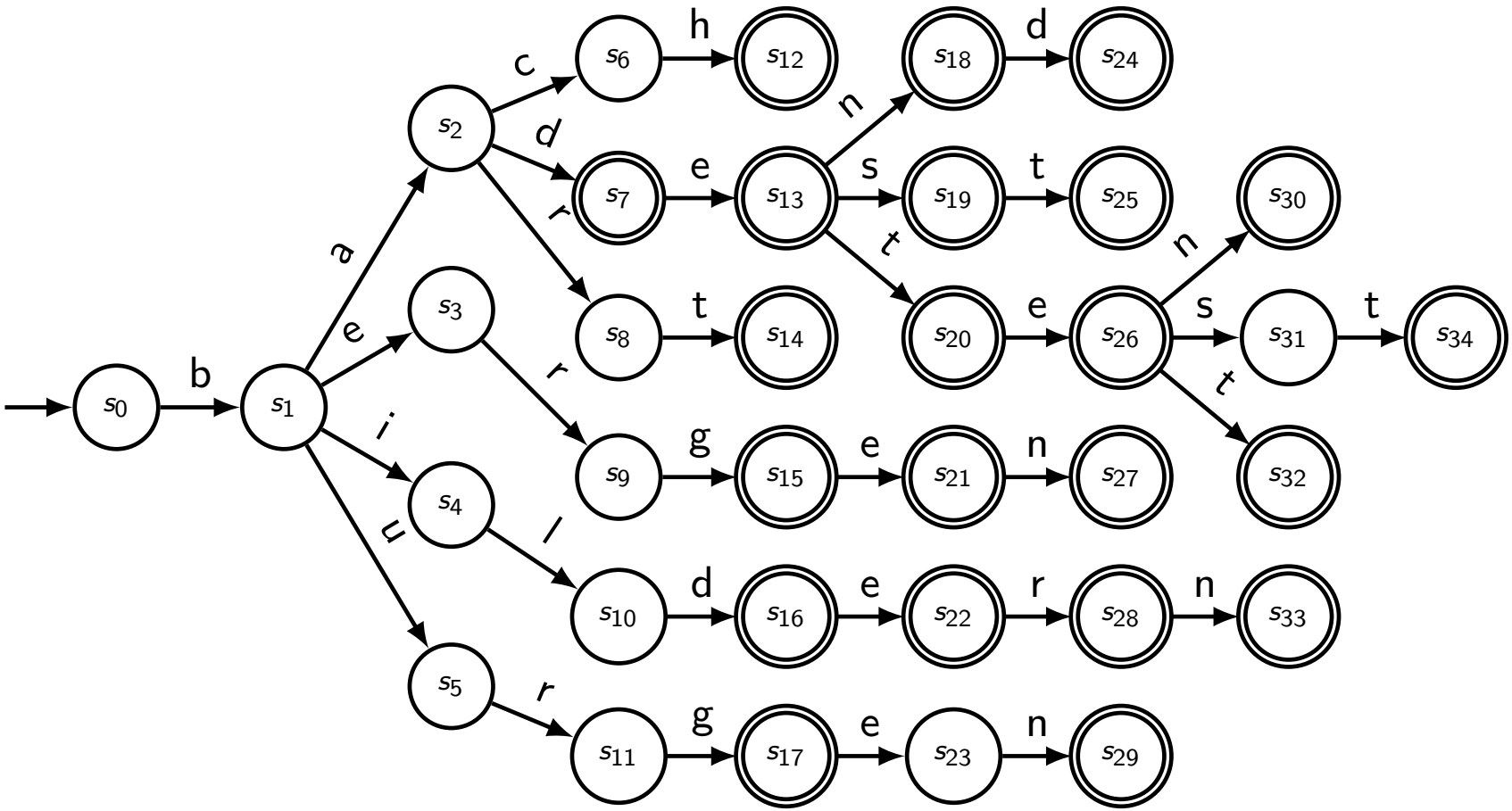


Tries as FSAs



- Moore or Mealy automaton?
- deterministic/nondeterministic?
- minimal?

Tries as FSAs



Tries as FSAs

Finite state automata (FSAs)

- accept/generate regular languages
- can be decribed by means of regular expressions

```
b(a(ch\ )|                                    \
    d(\ |e(\ |                                \
          (n(\ |d\ ))|                        \
          st\ |                               \
          t(\ |e(\ |n\ )|st\ |t\ )))))|      \
  erg(\ |e(\ |n\ ))|                          \
  ild(\ |e(\ |r(\ |n\ )))|                    \
  urg(\ |e(\ |n\ )))
```

Tries as FSAs

- possible abstraction: name and reuse partial automata

```
$verb_dt_infl$ := e(\ |(n\ |st\ |t\ )
$verb_dt_past$ := (et\ )
$verb_dt_root$ = bad | rett | leit | leid | ...
```

- concatenation of two automata to form a new one

```
$present_tense_forms$ = $verb_dt_root$ $verb_dt_infl$
$past_tense_forms$    = $verb_dt_root$ $verb_dt_past$ \
                            $verb_dt_infl$
```

`present_tense_form`	*bade*, *baden*, *badest*, *badet*
`past_tense_form`	*badete*, *badeten*, *badetest*, *badetet*

Tries as FSAs

- regular languages are closed under
  - union ($A \bigcup B, A|B$)
  - concatenation (AB)
  - transitive closure ($A*$)
  - intersection (A $\bigcap$ B)
  - difference ($A - B$)
  - complementation ($\bar{A}$)
  - string reversal (A^R)
- algebra for calculating with automata

```
$verb_forms$ = \
    $present_tense_forms$ | $past_tense_forms$
```

- creating a model consists of two repatedly executed steps
  - combine FSAs
  - minimize the resulting FSA

Probabilistic Tries

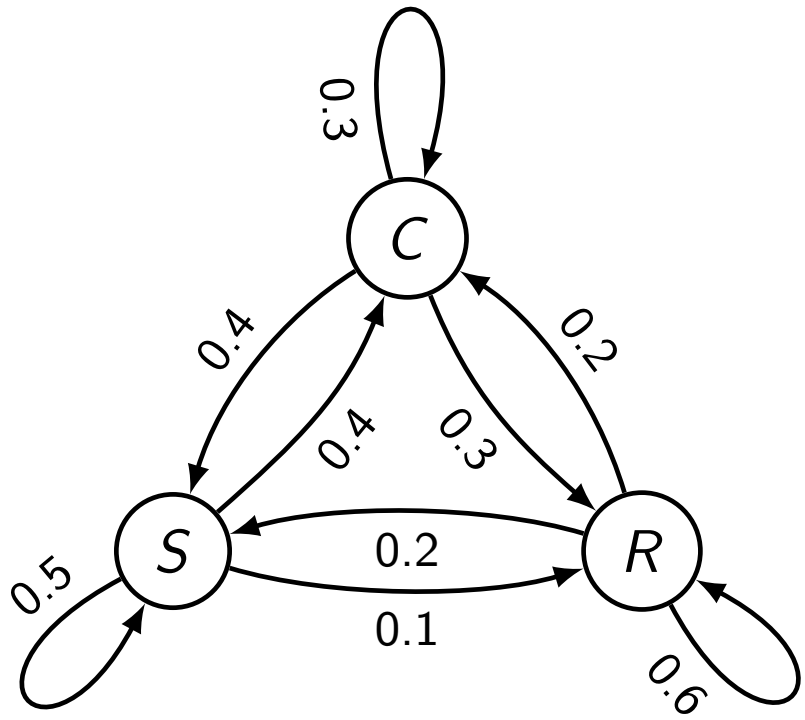
- automaton can be extended with edge weights
- a weighted finite-state automaton (wFSA) is an FSA with three scoring functions:
  - initial output function which assigns a weight to the initial state of the automaton, the initial probability $P(s)$
  - an output function which assigns a weight to transitions in the automaton, e.g. transition probabilities $P(s_i|s_{i-1})$
  - a final output function which assigns a weight to leaving the automaton, e.g. the exit probabilities of the final states $P(s_i), s_i \in F$
- if the weights are probabilities, they must sum to one for all edges leaving a node

Markov models

- special case of a weighted FSA: Markov chain, Markov model
- state-based model
  - states $S = s_1, \ldots, s_n$
  - state-transition probabilities $P(s_i|s_j)$

Markov models

- example: Markov model for weather prediction



	sunny	cloudy	rainy
sunny	0.5	0.4	0.1
cloudy	0.4	0.3	0.3
rainy	0.2	0.2	0.6

Markov models

- Markov assumption: the state transition probability only depends on the current state

$$P(s_{t+1}|s_0 \ldots s_t) = P(s_{t+1}|s_t).$$

- thus, s_t conveys all the information about the history that can affect the future:

> "The future is independent of the past given the present."

Markov Models

- additional assumption: stationary Markov process
- for all $t > 0$, $t' > 0$,

$$P(s_{t+1}|s_t) = P(s_{t'+1}|s_{t'}).$$

- only $P(s_0)$ and $P(s_{t+1}|s_t)$ need to be specified
  - simple model, easy to train
  - often the most natural model
  - the network can extend indefinitely
- states uniquely correspond to observations

Markov Models

Examples of Markov chains for German letter sequences

unigrams:	aiobnin*tarsfneonlpiitdregedcoa*ds*e* dbieastnreleeucdkeaitb*dnurlarsls*omn* keu**svdleeoieei* ...
bigrams:	er*agepteprteiningeit*gerelen*re*unk* ves*mterone*hin*d*an*nzerurbom* ...
trigrams:	billunten*zugen*die*hin*se*sch*wel*war* gen*man*nicheleblant*diertunderstim* ...
quadrograms:	eist*des*nich*in*den*plassen*kann*tragen* was*wiese*zufahr* ...

Probabilistic Tries

- Is a probabilistic trie a stationary Markov model?

Probabilistic Tries

- Is a probabilistic trie a stationary Markov model?
- possible predictions produced by a probabilistic trie
  - the most probable next character following an initial string of a word:

$$P(c_{i+1} \mid c_1 \ \ldots \ c_i) =$$

Probabilistic Tries

- Is a probabilistic trie a stationary Markov model?
- possible predictions produced by a probabilistic trie
  - the most probable next character following an initial string of a word:

$$P(c_{i+1} \mid c_1 \ \ldots \ c_i) = P(c_{i+1} \mid c_i)$$

Probabilistic Tries

- Is a probabilistic trie a stationary Markov model?
- possible predictions produced by a probabilistic trie
  - the most probable next character following an initial string of a word:

$$P(c_{i+1} \mid c_1 \ \ldots \ c_i) = P(c_{i+1} \mid c_i) \qquad ???$$

Probabilistic Tries

- Is a probabilistic trie a stationary MARKOV model?
- possible predictions produced by a probabilistic trie
  - the most probable next character following an initial string of a word:

    $$P(c_{i+1} \mid c_1 \ \ldots \ c_i) = P(c_{i+1} \mid c_i) \qquad ???$$

    Looks like a MARKOV assumption but isn't. Why?

Probabilistic Tries

- Is a probabilistic trie a stationary MARKOV model?
- possible predictions produced by a probabilistic trie
  - the most probable next character following an initial string of a word:

    $$P(c_{i+1} \mid c_1 \ \ldots \ c_i) = P(c_{i+1} \mid c_i) \qquad ???$$

    Looks like a MARKOV assumption but isn't. Why?

    $$P(c_{i+1} \mid e(s_1) = c_1 \ \ldots \ e(s_j) = c_i)$$
    $$= P(e(s_{j+1}) = c_{i+1} \mid e(s_j) = c_i)$$

Probabilistic Tries

- Is a probabilistic trie a stationary MARKOV model?
- possible predictions produced by a probabilistic trie
  - the most probable next character following an initial string of a word:

    $$P(c_{i+1} \mid c_1 \ \ldots \ c_i) = P(c_{i+1} \mid c_i) \qquad ???$$

    Looks like a MARKOV assumption but isn't. Why?

    $$P(c_{i+1} \mid e(s_1) = c_1 \ \ldots \ e(s_j) = c_i)$$
    $$= P(e(s_{j+1}) = c_{i+1} \mid e(s_j) = c_i)$$

- first order Markov model for states, but not for characters!

Probabilistic Tries

- possible predictions (cont.)
  - the probability of a word or the initial string of a word

$$P(c_1 \ldots c_n)$$

$$= P(e(s_1) = c_1)$$

$$\cdot \prod_{i=2}^{n} P(e(s_j) = c_i \mid e(s_{j-1}) = c_{i-1})$$

Probabilistic Tries

- possible predictions (cont.)
  - the probability of a word or the initial string of a word

$$P(c_1 \ldots c_n)$$

$$= P(e(s_1) = c_1)$$

$$\cdot \prod_{i=2}^{n} P(e(s_j) = c_i \mid e(s_{j-1}) = c_{i-1})$$

  - the probability of a word completion

$$P(c_{m+1} \ldots c_n \mid e(s_1) = c_1 \ \ldots \ e(s_j) = c_m)$$

$$= \prod_{i=m}^{n} P(e(s_{j+1}) = c_{i+1} \mid e(s_j) = c_i)$$

Probabilistic Tries

- the probability of the same completion can be different for different initial strings

$$w_1 = a_1 \ldots a_k \ldots a_m \wedge w_2 = b_1 \ldots b_l \ldots b_n$$
$$\wedge\, a_{k+1} \ldots a_m = b_{l+1} \ldots b_n \wedge a_1 \ldots a_k \neq b_1 \ldots b_l$$
$$\neg \rightarrow P(a_{k+1} \ldots a_m) = P(b_{l+1} \ldots b_n)$$

- the probabilities can be estimated on corpus data
- very simple case of a machine learning technique
  - a formal framework with a set of free parameters (probabilities)
  - optimal parameter settings are determined on sample data
- a general model of character sequences can be interpolated with a user-specific one

Probabilistic Tries

- predictions can be made incrementally
  - the probabilities can be updated as soon as new evidence (the next character) becomes available
  - no necessity to wait until the end of the string
  - the previous prediction can be reused to compute the current one
  - (spoken) language evolves over time
    - it is produced and perceived incrementally
- incremental language processing is a major benefit in interactive applications
  - rapid response capability
  - quick correction possibilities
  - intuitive human-computer interfaces

String similarity

- similarity/dissimilarity of two character strings can be measured by means of elastic string matching
- applications for
  - spelling correction
  - robust data base access, tolerates spelling errors
- based on the optimal alignment of the two sequences
  - optimization task
  - can be efficiently solved by means of dynamic programming
- the principle of dynamic programming:
  "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." (Bellman, 1957)

String similarity

- optimization requires a local distance metric
  - most simple case: string edit distance (LEVENSHTEIN metric)

    $c(\text{match}) = 0, c(\text{mismatch}) = 1$

    $$\text{with mismatch} \in \{\text{insertion,deletion,substitution}\}$$

- a string A which requires less repair operations than B to reconstruct C is considered more similar to C than B.
- more sophisticated cost functions can capture additional domain knowledge

String similarity

- optimization requires a local distance metric
  - most simple case: string edit distance (LEVENSHTEIN metric)

    $c(\text{match}) = 0, c(\text{mismatch}) = 1$

    $$\text{with mismatch} \in \{\text{insertion,deletion,substitution}\}$$

- a string A which requires less repair operations than B to reconstruct C is considered more similar to C than B.

- more sophisticated cost functions can capture additional domain knowledge
  - neighbourhood on a keyboard
  - phonetic similarities
  - user specific confusions
  - ...

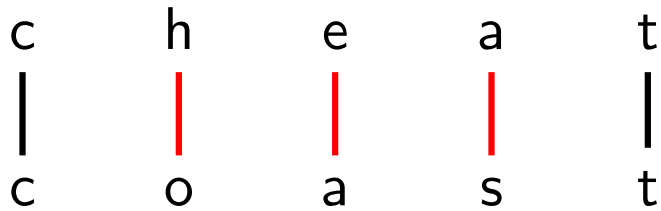
String similarity

Alternative alignments with the same distance are possible

c	h	e	a	t
c	o	a	s	t

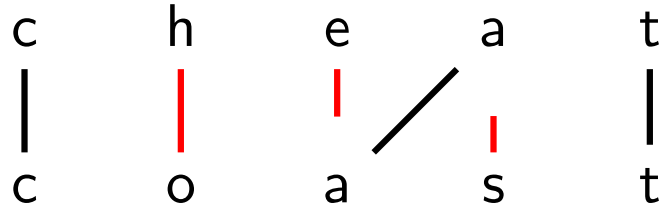
String similarity

Alternative alignments with the same distance are possible



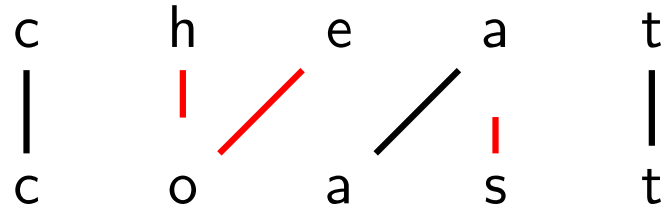
String similarity

Alternative alignments with the same distance are possible



String similarity

Alternative alignments with the same distance are possible



String similarity

- Representation of search states

$$\langle position\_in\_A, position\_in\_B, costs \rangle$$

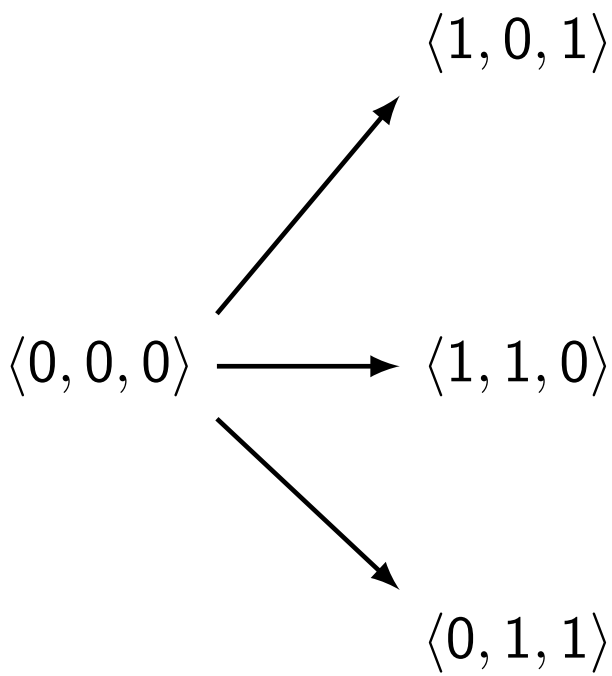
- State transitions

$$\langle i, j, c_{old} \rangle \Rightarrow \begin{cases} \langle i+1, j+1, c_{new} \rangle & c_{new} = \begin{cases} c_{old} & if \quad a_i = b_j \\ c_{old}+1 & else \end{cases} \\ \langle i+1, j, c_{old}+1 \rangle \\ \langle i, j+1, c_{old}+1 \rangle \end{cases}$$

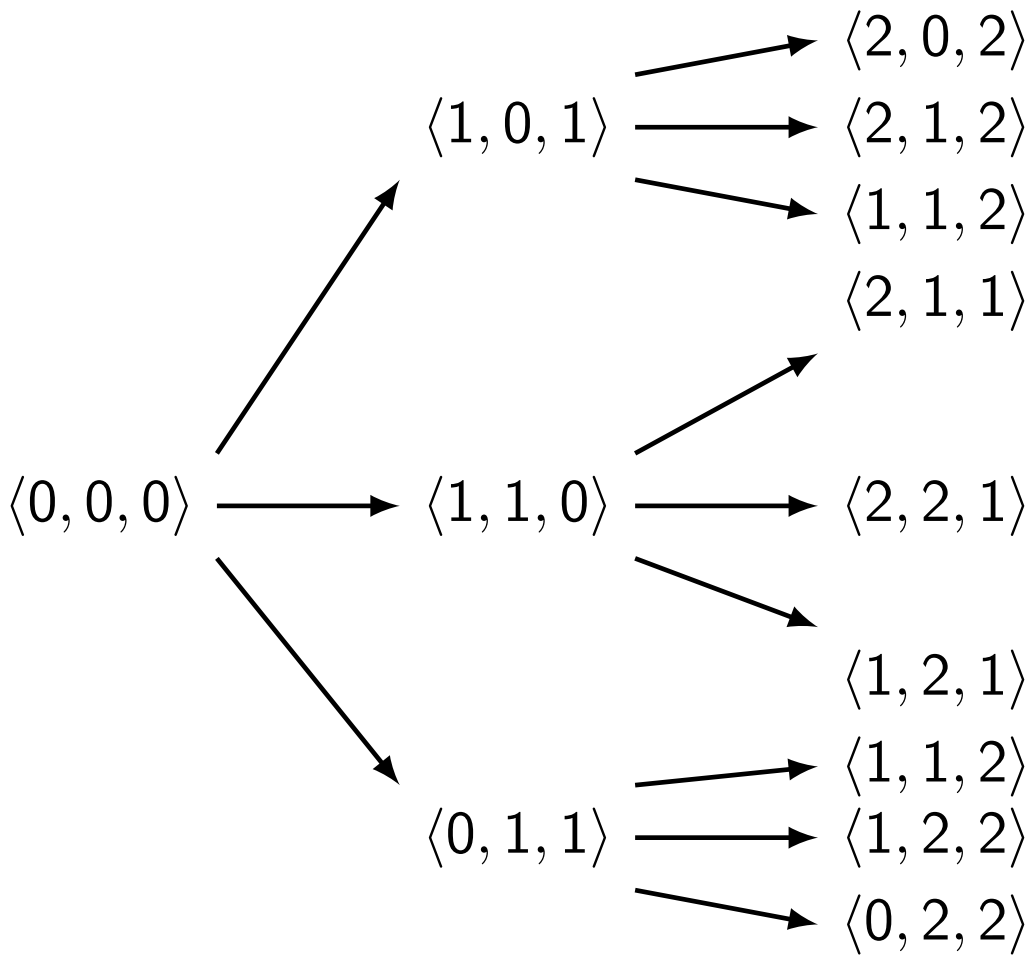
String similarity

$\langle 0, 0, 0 \rangle$

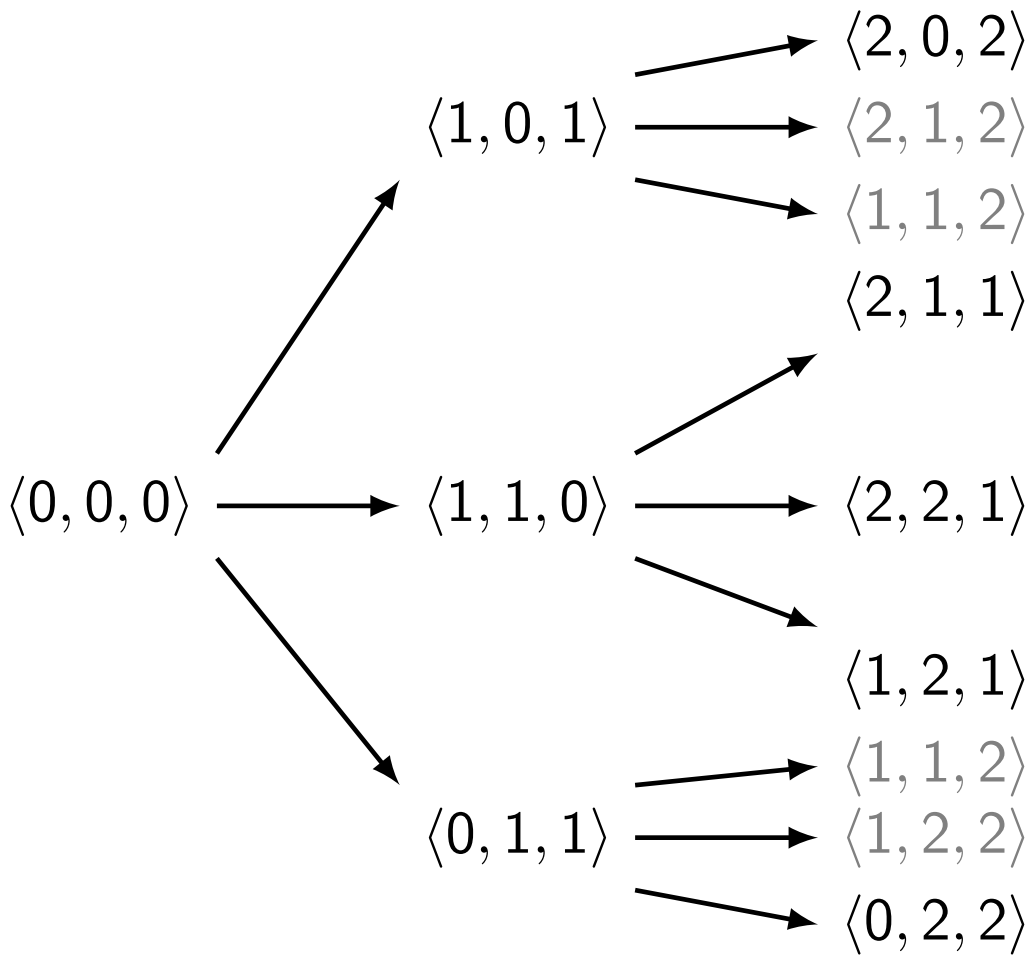
String similarity



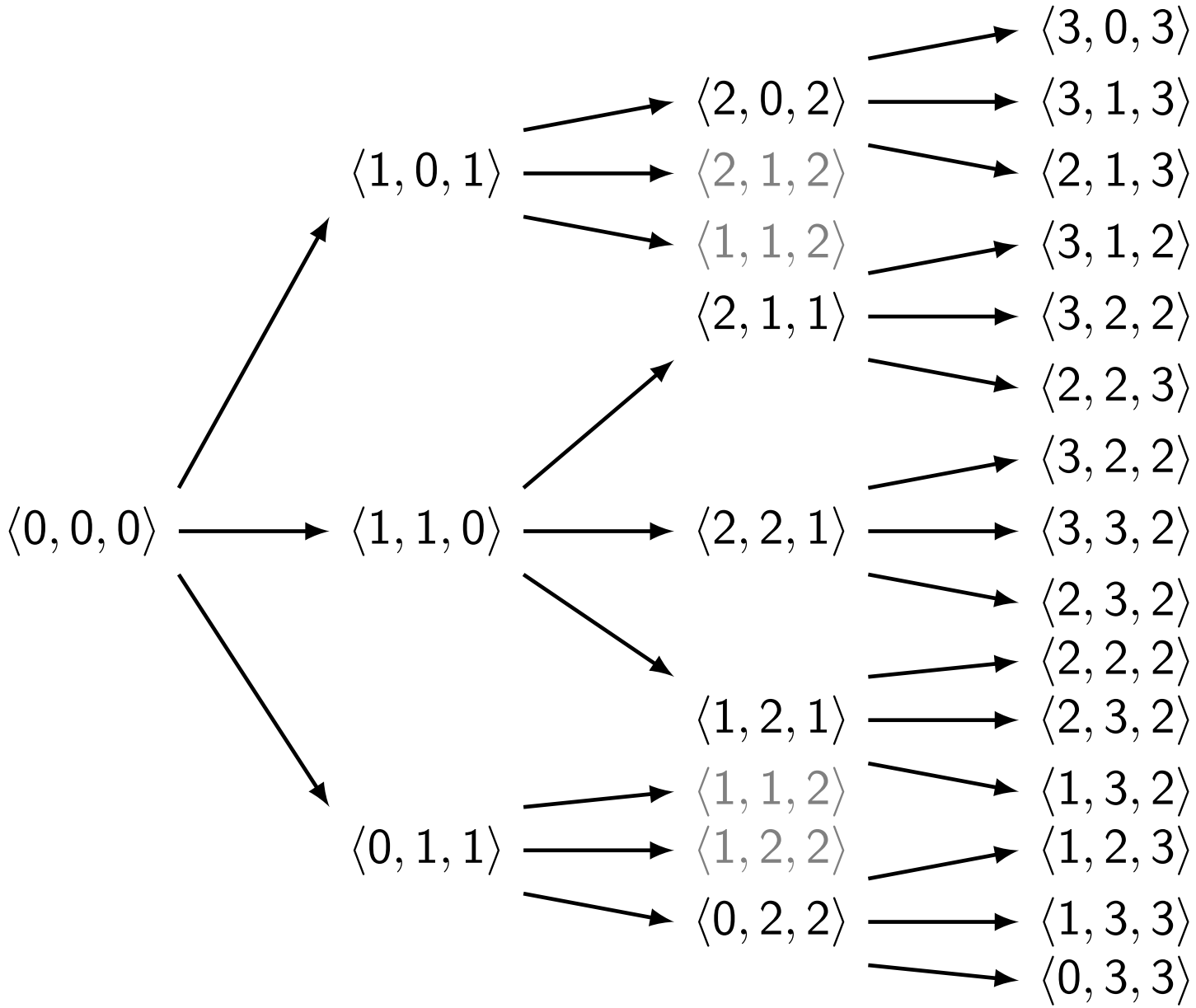
String similarity



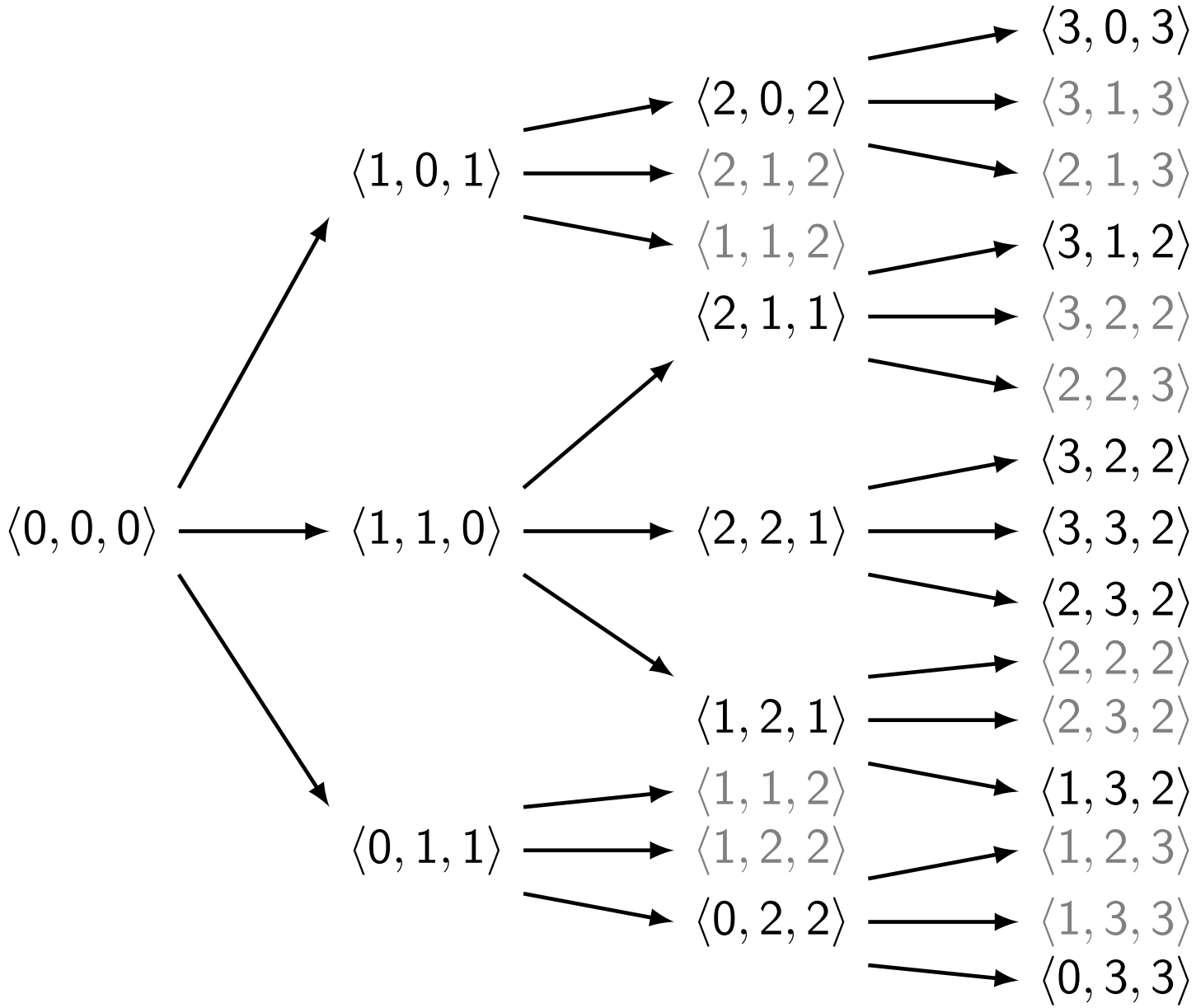
String similarity



String similarity



String similarity



String similarity

Populating the distance table

local distances

		c	h	e	a	t
	0	1	1	1	1	1
c	1	0	1	1	1	1
o	1	1	1	1	1	1
a	1	1	1	1	0	1
s	1	1	1	1	1	1
t	1	1	1	1	1	0

global distances

String similarity

Populating the distance table

local distances

		c	h	e	a	t
	0	1	1	1	1	1
c	1	0	1	1	1	1
o	1	1	1	1	1	1
a	1	1	1	1	0	1
s	1	1	1	1	1	1
t	1	1	1	1	1	0

global distances

		c	h	e	a	t
	0	1	2	3	4	5
c	1					
o	2					
a	3					
s	4					
t	5					

String similarity

Populating the distance table

local distances

		c	h	e	a	t
	0	1	1	1	1	1
c	1	0	1	1	1	1
o	1	1	1	1	1	1
a	1	1	1	1	0	1
s	1	1	1	1	1	1
t	1	1	1	1	1	0

global distances

		c	h	e	a	t
	0	1	2	3	4	5
c	1	0	1	2	3	4
o	2	1	1	2	3	4
a	3	2	2	2	2	3
s	4	3	3	3	3	3
t	5	4	4	4	4	3

String similarity

Populating the distance table

local distances

		c	h	e	a	t
	0	1	1	1	1	1
c	1	0	1	1	1	1
o	1	1	1	1	1	1
a	1	1	1	1	0	1
s	1	1	1	1	1	1
t	1	1	1	1	1	0

global distances

		c	h	e	a	t
	0	1	2	3	4	5
c	1	0	1	2	3	4
o	2	1	1	2	3	4
a	3	2	2	2	2	3
s	4	3	3	3	3	3
t	5	4	4	4	4	3

Words and Wordforms

- Lexical items
- Dictionary lookup
- Word segmentation
- Morphological analysis
- Morphophonology
- Lexical semantics
- Distributed representations
- Part-of-speech tagging
- Word-sense disambiguation

Words and Wordforms

- Lexical items
- Dictionary lookup
- Word segmentation
- Morphological analysis
- Morphophonology
- Lexical semantics
- Distributed representations
- Part-of-speech tagging
- Word-sense disambiguation

Word segmentation

- Combinatorial word segmentation
- Word segmentation with Markov models
- Unsupervised word segmentation
- Applications of word segmentation

Word segmentation

- Combinatorial word segmentation
- Word segmentation with Markov models
- Unsupervised word segmentation
- Applications of word segmentation

Combinatorial word segmentation

- splitting a wordform into its morphological constituents:
- roots and affixes
  - only for concatenative morphology
- syllables
  - hyphenation for print and word processing

Combinatorial word segmentation

- not always morphs = syllables

  tausch-en vs. *tau-schen*

  but prefix and compound boundaries are usually good break points

- not always spoken = written syllables

  ba-cken vs. *bak-ken*

- segmentation often considered the core of morphology:

  "learning morphology"
  - neglecting non-concatenative phenomena
  - neglecting the linguistic meaning of morphemes, i.e. the syntactic and semantic consequences of morphological derivations

Combinatorial word segmentation

- tasks of different degrees of difficulty
  - fully informed: complete morph inventory known
  - partly informed: only affixes known, not root inventory
    - set of affixes is largely domain independent
    - set of roots is not
  - unsupervised machine learning: neither roots nor affixes known

Combinatorial word segmentation

- fully informed word segmentation: combinatorial decomposition

  Ein-ge-ständ-nis
  Unter-such-ung-s-kommiss-ion

- rare cases of true ambiguity:

  Wacht-raum/Wach-traum

- spurious ambiguities:

  ung-er-n (S-S-S), *lich-t-er* (S-S-P/S)
  T-e-ig (S-S-S), *Bar-bar-a* (S-S-P), *S-t-ie-ge* (S-S-S-P)
  T-a-s-t-e (S-P-S-S-S)

- affixes are bound morphemes
  can only appear together with a free one

Combinatorial word segmentation

- "grammar"-based segmentation
  - licensing admissible morpheme sequences by means of a finite state automaton

    ```
    ($Prefix$* $Root$ $Suffix$*)*
    ```

- grammar-based segmentation without a root dictionary produces spurious ambiguities
  ,
  - e.g. prefix segmentation: stortest/longest match first?

    *a: a-moralisch, *a-mortisieren*
    *ab: ab-fahren, *ab-ort, *ab-iologisch*
    *aber: aber-glaube, *aber-kennen, *aber-nten*

    phonological constraint: *nt* is impossible as initial consonant cluster

Combinatorial word segmentation

- approximation of a root inventory by means of a finite state automaton

```
$Root$ = $C$* $V$ $C$*
$C$    = b | c | d | f | g | h ... y | z
$V$    = a | e | i | o | u | ä | ö | ü | y
```

- refining the model: only phonologically possible initial/final consonant clusters considered

```
$Root$      = $CCinitial$ V $CCfinal$

$CCinitial$ = bl | br | ch | chl | chr | d | dr |  \
              ... x | y | z

$CCfinal$   = b | lb | rb | bd | dd | hd | ld | md |  \
              nd | rd | ... | x | y | z | rz | tz
```

Word segmentation

- Combinatorial word segmentation
- Word segmentation with Markov models
- Unsupervised word segmentation
- Applications of word segmentation

Word segmentation

- Combinatorial word segmentation
- Word segmentation with MARKOV models
- Unsupervised word segmentation
- Applications of word segmentation

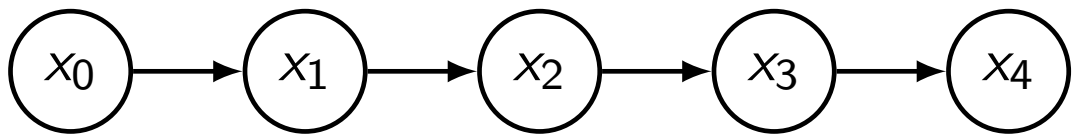
Markov models

- modeling admissible morpheme sequences and root approximations by means of a probabilistic "language model"
  - states represent morphs
  - state-transitions model morph concatenation
- assumption: stationary Markov model

Higher-order Markov Models

- higher order dependencies cannot be represented in a state-transition diagram

  → alternative graphical representation as probabilistic dependency network (Bayesian network)



- each node represents a conditional probability distribution

$$P(x_n|x_{n-1}) = \begin{pmatrix} P(x_n = m_i|x_{n-1} = m_j) \end{pmatrix} \qquad i, j = 1, ..., k$$

  k: # states (different observations/morphs)

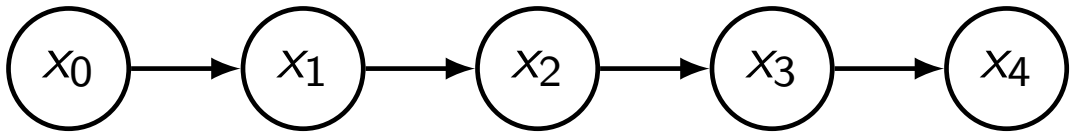
- all nodes share the same probability distribution
- the model is "rolled out" to the desired length

Higher-order Markov models

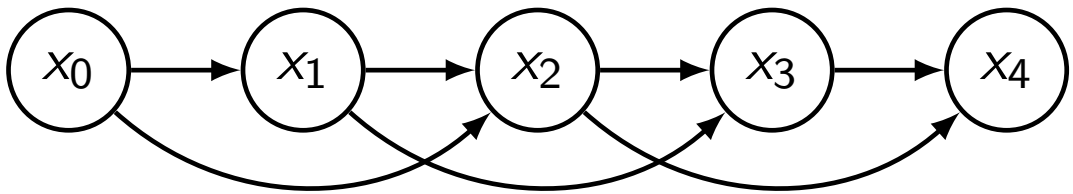
	Bayesian network	State transition diagram
nodes	variables with states as values	states
edges into nodes	causal influences and their probabilities	possible state transitions
# nodes	length of the observation sequence	# model states

Higher-order Markov Models

- dependencies of different length can be modelled
- bigrams



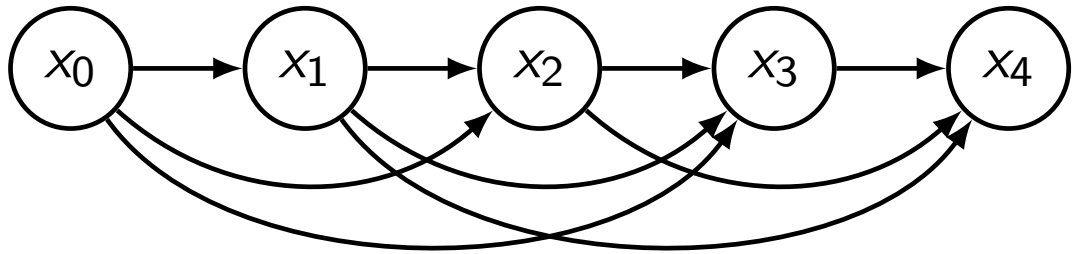
- trigrams



  - three different time slices have to be modelled
    - for variable/node x_0: $P(x_0 = s_r)$
    - for variable/node x_1: $P(x_1 = s_i | x_0 = s_j)$
    - for all other variables/nodes x_n:
      $$P(x_n = s_i | x_{n-2} = s_j, x_{n-1} = s_k)$$

Higher-order Markov Models

- quadrograms: $P(s_i|s_{i-3}\, s_{i-2}\, s_{i-1})$



- four different kinds of time slices required
  - for variable/node x_0: $P(x_0 = s_i)$
  - for variable/node x_1: $P(x_1 = s_i|x_0 = s_j)$
  - for variable/node x_2: $P(x_2 = s_i|x_0 = s_j,\, x_1 = s_k)$
  - for all other variables/nodes x_n:
    $$P(x_n = s_i|x_{n-3} = s_j,\, x_{n-2} = s_k,\, x_{n-1} = s_l)$$

Markov Models

- Markov models provide additional information about likely morpheme sequences
  - a grammar can only enumerate the alternatives
  - a probabilistic model can weight and rank them
    $\rightarrow$ preferential reasoning

Markov Models

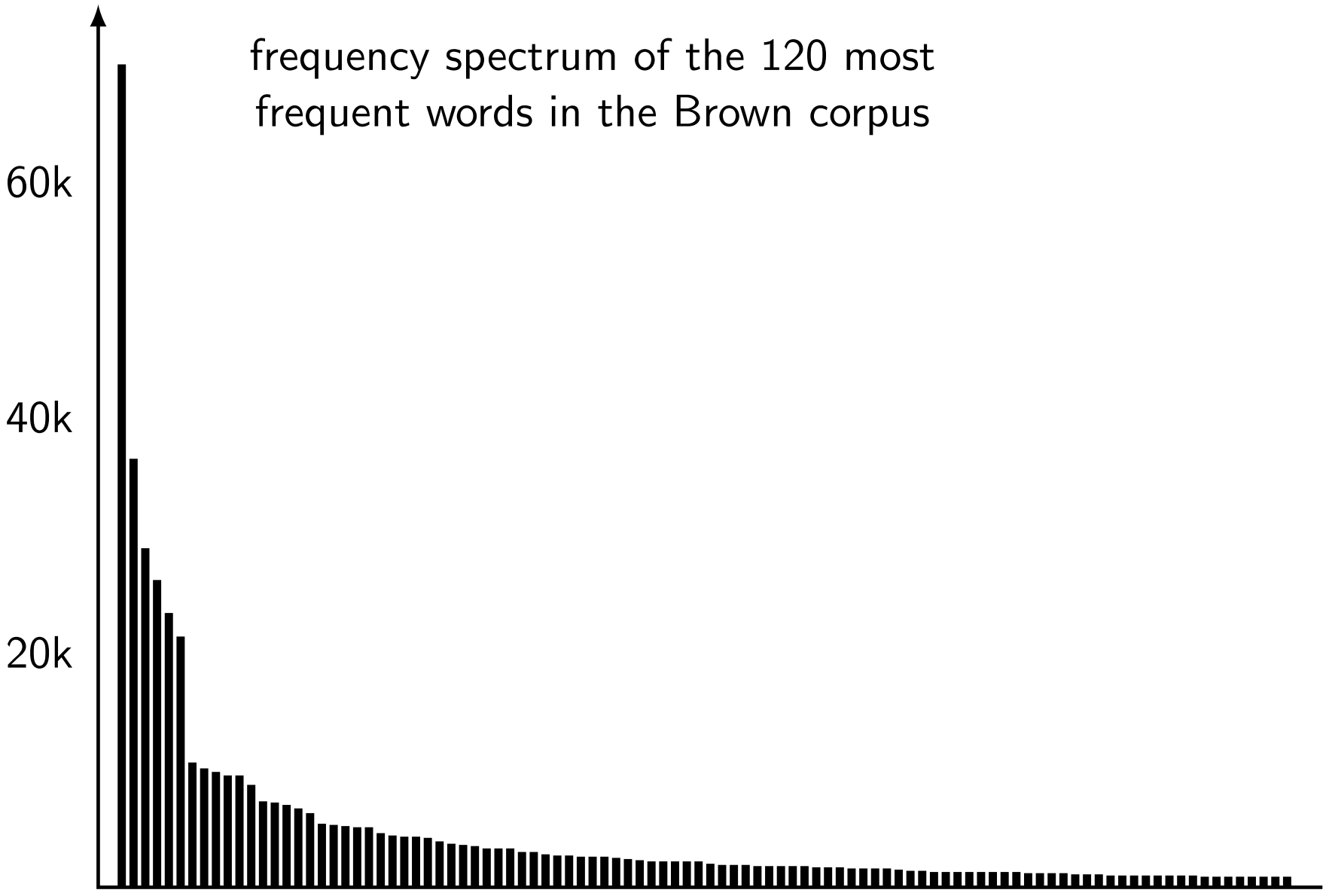
- Markov models provide additional information about likely morpheme sequences
  - a grammar can only enumerate the alternatives
  - a probabilistic model can weight and rank them
    $\rightarrow$ preferential reasoning
- probabilistic models are particularly useful, if the probability distributions are skewed
  - most distributions in natural language follow roughly Zipfs law:

$$f(w) \cdot r(w) = const$$

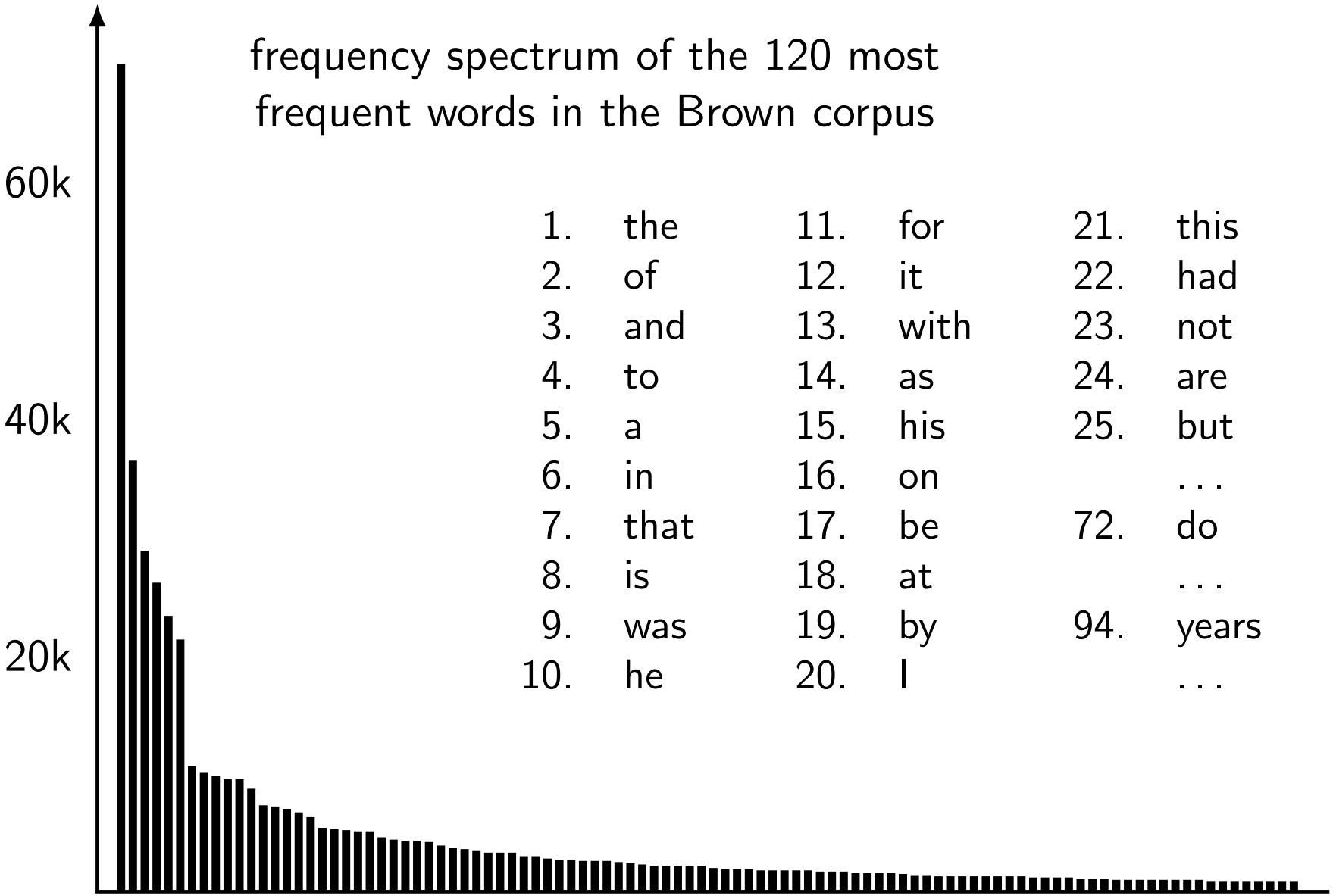
    - few high frequency items
    - overwhelmingly many low frequency items

  $\rightarrow$ probabilistic models can contribute a large amount of additional information

Markov Models



Markov Models



Markov Models

problems with probabilistic models

- rare events
  - the probability distributions have to be estimated/trained on data
  - most of the combinations will never been seen during training
  - hapax legomena become a stochastic phenomenon
  - irrespective how large the corpus is
    - smoothing and backoff techniques can be applied
- lack of appropriate data
  - the data has to be properly annotated
  - annotation requires huge human effort
  - are annotations really necessary?

Word segmentation

- Combinatorial word segmentation
- Word segmentation with Markov models
- Unsupervised word segmentation
- Applications of word segmentation

Word segmentation

- Combinatorial word segmentation
- Word segmentation with MARKOV models
- Unsupervised word segmentation
- Applications of word segmentation

Unsupervised word segmentation

- tries to find the optimal segmentation of a corpus
- optimization criterion: minimal description length (MDL)

  "The number of letters in a list of word is greater than the number of letters in a list of stems and affixes that are present in the original list." Goldsmith (2001)
- naïve description length:

$$\left\{ \begin{array}{ccc} \textit{laughed} & \textit{laughing} & \textit{laughs} \\ \textit{walked} & \textit{walking} & \textit{walks} \\ \textit{jumped} & \textit{jumping} & \textit{jumps} \end{array} \right\} \quad \text{vs.} \quad \left\{ \begin{array}{c} \textit{laugh} \\ \textit{walk} \\ \textit{jump} \end{array} \right\} \left\{ \begin{array}{c} \textit{ed} \\ \textit{ing} \\ \textit{s} \end{array} \right\}$$

57 characters vs. 19 characters

Unsupervised word segmentation

- MDL-learning/inference as data compression

  "The MDL Principle is based on the following insight: any regularity in a given set of data can be used to compress the data, i.e. to describe it using fewer symbols than needed to describe the data literally. The more regularities there are, the more the data can be compressed." (Grünwald, 1998/2007)

  - D: data to be modelled (a corpus or a list of wordforms)
  - M: a model (the encoding scheme)
  - $L(M)$: length (in bits) of the description of the model
  - $L(D|M)$: length (in bits) of the data when encoded by means of M

  - task: find the optimal M by minimizing $L(M) + L(D|M)$

Unsupervised word segmentation

- the two components of the objective function are antagonistic
  - a trivial model M_1 (e.g. a full form list) can describe/encode the corpus perfectly
    - assigns a high probability $P(D|M_1)$ which corresponds to a low description length $L(D|M_1)$ (in an optimal code)
    - but is quite expensive in terms of its own description length $L(M_1)$
  - the optimal morphology M_2 has a lower probability for the corpus data (it commits errors)
    - requires a higher description length $L(D|M_1)$
    - but is itself cheaper in terms of description length $L(M_2)$
- MDL favors simple models with high adequacy

Unsupervised word segmentation

- idea first implemented in Linguistica Goldsmith (2001,2006)
- simplifications
  - compares only models with the same parametric complexity
  - focusses on concatenative morphology
  - restricts segmentation to two-part splits (e.g. stem + suffix or prefix + stem)
- simple concatenative model (morphotactics)
  - list of suffixes
  - list of stems
  - mapping between stems and suffixes (signatures)

Unsupervised word segmentation

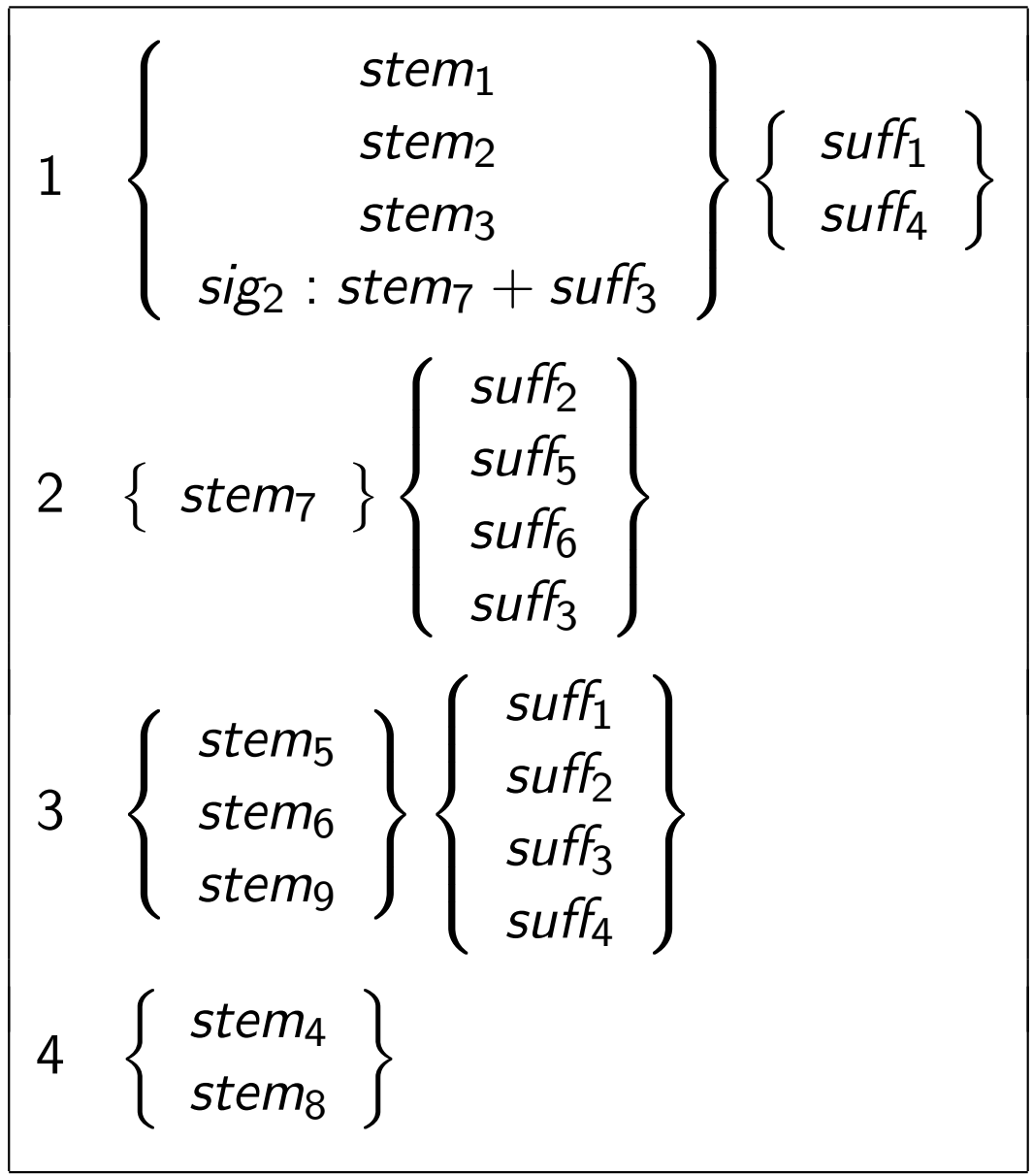
Suffixes

1	$\emptyset$
2	*ed*
3	*ing*
4	*s*
5	*e*
6	*es*

Stems

1	*cat*
2	*dog*
3	*hat*
4	*John*
5	*jump*
6	*laugh*
7	*sav*
8	*the*
9	*walk*

Signatures



Unsupervised word segmentation

- description length of the model

$$L(M) = L(\text{Stem-Table}) + L(\text{Suffix-Table}) + L(\text{Signature-Table})$$

  - table length
  - item length
  - pointer length
  - ...

- description length of the data encoded by the model

$$\begin{aligned} L(D|M) &= -\sum_{w \in Corpus} \log P(w) \\ &= -\sum_{w \in Corpus} \log P(sig) + \log P(stem|sig) \\ &\quad + \log P(suffix|sig) \end{aligned}$$

Unsupervised word segmentation

- two phases of the algorithm

1. finding good initial splitting points
   - guided by transition frequencies
2. modifying the splitting points
   - guided by the entropy of the stem final characters (up to 4)
   - reevaluating these modifications with respect to the description length

- tested on the first 200,000 and the first 300,000 words of the Brown corpus
  - segmentation accuracy 72%
  - 30% of the errors due to inaccurately cutting off stem-final *-e*

Unsupervised word segmentation

- Morfessor (Creutz and Lagus 2005)
  - number of segments is not restricted
  - maximum aposteriori model instead of MDL
  - results for English comparable to Linguistica
  - considerably better results for Finnish (many splits per wordform)

Word segmentation

- Combinatorial word segmentation
- Word segmentation with MARKOV models
- Unsupervised word segmentation
- Applications of word segmentation

Word segmentation

- Combinatorial word segmentation
- Word segmentation with MARKOV models
- Unsupervised word segmentation
- Applications of word segmentation

Applications of word segmentation

- hyphenation
- text mining
  - lemmatization
  - stemming
- text-to-speech synthesis
- morpheme-based language models
- linguistic field studies

Applications: Hyphenation

- compound boundaries are always very good splitting points
- prefix boundaries are mostly good splitting points (only exception: *a*)
- suffix boundaries help to find good splitting points

  täu-schen vs. Häus-chen

- sometimes influenced by pragmatic factors

Applications: Hyphenation

- high baseline even without morphological support:
  - the last consonant goes to the next line
  - yields already $\approx 90\%$ accuracy
- but heuristics fails most often at (undetected) compound boundaries

Haus-bau	*but:*	**Hausf-lur, *Hau-sein-gang*
Haut-farbe	*but:*	**Hautc-reme, *hau-teng*
Bau-zaun	*but:*	**Baus-telle, *Bauar-beiter*
Baum-rinde	*but:*	**Baums-tamm, *Bau-maffe, *Bau-mart*
	but:	*Bau-markt*

Applications: Lemmatization/Stemming

- text mining: dictionary access, information retrieval, information extraction, question answering, topic modeling, ...
- lemmatization: determining a canonical form

  baue, baust, baut, bauen, baute, bautest, bauten, bautet
  $\rightarrow$ *bauen*

- simplified task: stemming (stripping off inflectional endings)

  bau-e, bau-st, bau-t, bau-en, bau-te, bau-t-est, bau-t-en, bau-t-et
  $\rightarrow$ *bau-*

Applications: Lemmatization/Stemming

- problems with simple stemming approaches
  - phonological change

    easy $\rightarrow$ *easi-ly*
    dependen-cy $\rightarrow$ *dependen-cie-s*
  - stem inflection for strong verbs

    trage, trägst, trägt, tragen, tragt, trug, trugst, trugen, trugt
    $\rightarrow$ *tragen*
  - stem inflection with plural nouns

    Apfel, Äpfel $\rightarrow$ *Apfel*
    Baum, Bäume $\rightarrow$ *Baum*

    but:
    Säge, Sägen $\rightarrow$ *Säge*
    Bürste, Bürsten $\rightarrow$ *Bürste*

Applications: Text-to-speech synthesis

- pronunciation often depends on the word internal structure
  - final devoicing (*leid-lich*)
  - splitting into speech syllables (*Rand-erscheinung* vs. *Rander-scheinung*)
  - vowel quality (*Geb-en* vs. *Ge-burt*)
  - vowel clustering (*Elektroofen*)
  - consonant clustering (*Rös-chen* vs. *rö-schen*)
- sometimes pure word segmentation information is not sufficient
  - word stress/vowel quality (*Koffe-in* vs. *Bäuer-in*)

Applications: Morpheme-based language models

- language model: approximating the probability of an utterance (speech recognition, machine translation)

$$P(w_1 ... w_n) = P(w_1) \prod_{i=2}^{n} P(w_i | w_{i-1})$$

$$P(m_1 ... m_k) = P(m_1) \prod_{i=2}^{k} P(m_i | m_{i-1})$$

- even word segmentations with low quality might be helpful
  $\rightarrow$ language processing for underresourced languages

Applications: Linguistic field studies

- prestructuring empirical data
  - usually very few data available
- producing initial hypotheses
- expert revision is necessary anyhow