Natural Language Processing

Wolfgang Menzel

Department für Informatik
Universität Hamburg

Natural Language Processing

NLP is ...

... engineering + science

... linguistics + technology

Natural Language Processing

- Engineering:
  - How to build a system?
  - How to select a suitable approache/tool/data source?
  - How to combine different approaches/tools/data sources?
  - How to optimize the performance with respect to quality and resource requirements?
    - time, space, data, wo-/manpower
- Science:
  - Why an approach/tool/data source works/fails?
  - Why an approach/tool/data source A works better than B?

Natural Language Processing

- Linguistics:
  - What are suitable description levels for language?
  - What are the rules of a language?
  - How meaning is etsablished and communicated?
  - What have languages in common? How do they differ?
  - How languages can be learnt?
- Technology:
  - How an application problem can be solved?
    - Machine translation
    - Information retrieval
    - Information extraction
    - Speech recognition
  - Does linguistic knowledge help or hinder?

Examples

- ... are important to illustrate concepts and models
- but: The language problem
- Common ground: English
- me:
  - German
  - (Russian)
  - ((Polish))
- you:
  - Amharic
  - ...
  - ...

Doing research in NLP

- Motivation
- Problem definition
- Modelling/Implementation
- Evaluation
- Discussion

Doing research in NLP

- Motivation:
  - Why is the task important?
  - Has the task been addressed before? For other/similar languages?
  - Is it realistic to solve the task?
- Problem definition:
  - What kind of input data?
  - What kind of processing results are expected?
  - What level of quality (process/results) is needed?

Doing research in NLP

- Modelling/Implementation:
  - Which information needs to be captured by the model?
  - Which information is actually captured and how good?
  - Which variants of the approach can be devised? Which parameters need to be tuned?
  - Which information sources are available/need to be developed
    - corpora, annotated corpora, dictionaries, grammars, ...
  - Which algorithms are available to apply the model to a task?
  - What are their computational properties?

Doing research in NLP

- Evaluation:
  - How to measure the performance of a solution?
    - metrics, data, procedure
  - How good is the solution (compared to a baseline)?
  - What's the contribution of the different model components?
  - Which are the most promising system versions?
- Discussion:
  - Why the approach is superior/inferior to previous ones/to other versions of the system?
  - Which are the particular strengths of the approach, where are its limitations?

Doing research in NLP

- Applying a cyclic approach
  - redefine the task
  - choose another modelling approach
  - modify the solution / choose other parameter settings

Content of the course

Part 1: Non-deterministic procedures

- search spaces
- search strategies and their resource requirements
- recombination (graph search)
- heuristic search (Viterbi, A*)
- relationship between NLP and non-deterministic procedures

Content of the course

Part 2: Dealing with sequences

- Finite state techniques
- Finite state morphology
- String-to-string matching
- Speech recognition 1: DTW
- Speech recognition 2: Hidden-Markov-Models
- Tagging

Content of the course

Part 3: Dealing with structures

- Dependency parsing
- Phrase-structure parsing
- Unification-based grammars
- Constraint-based models (HPSG)

Part 1: Non-deterministic procedures

- non-determinism
- search spaces
- search strategies and their resource requirements
- recombination (graph search)
- heuristic search (Viterbi, A*)
- non-determinism and NLP

Non-determinism

An algorithm is swaid to be non-deteministic if local decisions cannot be uniquely made and alternatives have to be considered instead.

- (route) planning
- scheduling
- diagnosis

Search spaces

- a non-deterministic algorith spans a search space
- a search space can be represented as a directed graph
  - states (e.g. crossroads)
  - state transitions (e.g. streets)
  - initial state(s) (e.g. starting point)
  - final state(s), goal state(s) (e.g. destination)
- choice points: Branchings of the graph

Search spaces

- many different variants of search problems
  - one initial state / many initial states
  - one final state / many final states
    - one search result suffices vs. all of them need to be found (exhaustive search, computationally complete)
  - acyclic vs. cyclic graphs
  - final state is known vs. only properties of the final state are known
  - ...

Search strategies

- simplest case: the search space is unfolded into a tree during search
- the search space can be traversed in different orders → different unfoldings
- forward search vs. backward search
- depth-first vs. breadth-first

Search strategies

- resource requirements for tree search
- simplifying assumption: uniform branching factor at choice points
  - time vs. space
  - depth-first vs. breadth-first
  - best case vs. worst case vs. mean case
- termination conditions

Search strategies

- recombination: search paths which lead to the same state can be recombined (graph search)
- requires identification of search states
- simple, if unique identifiers available
- more complex, if startes are described by structures
- base-level effort vs. meta-level effort

Heuristic search

- so far important simplifying assumptions made
  - all transitions at a choice point are equally good
  - all final states are equally good
- usually not valid. e.g.
  - different street conditions (e.g. slope), different street lengths
  - differently distant/acceptable goal states (e.g. shops)
- search becomes an optimization problem, e.g.
  - find the shortest path
  - find the best goal state

Heuristic search

- computational approaches for optimum path problems: A*-search, Viterbi-search
- A*-search
  - requires the existence of a residual cost estimation (how far I am probably still away from the goal state?)
  - guarantees to find the optimum
  - well suited for metrical spaces
- Viterbi-search
  - recombination search which only considers promising state transitions
  - can be easily combined with additional pruning heuristics (beam search)

Non-determinism and NLP

- Why is non-determinism so important for natural language processing?
- ambiguity on all levels:
  - acoustic ambiguity
  - lexical ambiguity
    - homographs, homonyms, polysemie
  - morphological ambiguity
    - segmentation, syntactic function of morphs
  - syntactic ambiguity
    - segmentation, attachment, functional roles
  - semantic ambiguity
    - scopus
  - pragmatic ambiguity
    - question vs. answer

Part 2: Dealing with sequences

- Finite state techniques
- String-to-string matching
- Speech recognition 1: DTW
- Speech recognition 2: Hidden-Markov-Models
- POS-Tagging

Finite state techniques

- regular expressions
  - symbols: `a b c ...`
  - sequences of symbols: `abc xyz ...`
  - sets of alternative symbols `[abc] [a-zA-Z] ...`
  - complementation of symbols `[^a] [^ab] [^a-z]`
  - wildcard (any symbol): `.`
  - counter for symbols or expressions
    - none or arbitrary many: `a* [0-9]* .* ...`
    - at least one: `a+ [0-9]+ .+ ...`
    - none or one: `a? [0-9]? .? ...`
  - alternatives of expressions: (a*|b*|c*)

Finite state techniques

- Finite state automata
  - finite alphabet of symbols
  - states
  - start state
  - final state(s)
  - labelled (or unlabelled) transitions
- an input string is consumed symbol by symbol by traversing the automaton at transitions labelled with the current input symbol
- declarative model can be used for analysis and generation
- two alternative representations
  - graph
  - transition table

Finite state techniques

- Mapping between regular expressions and finite state automata
  - symbol $\rightarrow$ transition labeled with the symbol
  - sequence $\rightarrow$ sequence of transitions connected at a state (node)
  - alternative $\rightarrow$ parallel transitions or subgraphs connecting the same states
  - counter $\rightarrow$ transition back to the initial state of the subgraph or skipping the subgraph
  - wildcard: parallel transitions labelled with all the symbols from the alphabet
  - complementation: parallel transitions labelled with all but the specified symbols

Finite state techniques

- regular grammars
  - substitution rules of the type
    - $NT_1 \rightarrow NT_2$ T
    - NT $\rightarrow$ NT T
    - NT $\rightarrow$ T

    with NT is a non-terminal symbol and T is a terminal symbol

Finite state techniques

- regular expressions, finite state machines and regular grammars are three formalisms to describe regular languages
- they are equivalent, i.e. they can be transformed into each other without loss of model information

Finite state techniques

- deterministic FSA: each transition leaving a state carries another symbol
- non-deterministic FSA: else
- each FSA with an unlabelled transition is a non-deterministic one
- each FSA with unlabelled transitions can be transformed into an equivalent one without
- each non-deterministic FSA can be transformed into an equivalent deterministic one
  - additional states might become necessary

Finite state techniques

- composition of FSAs
  - concatenation: sequential coupling
  - disjunction/union: parallel coupling
  - repetition
  - intersection: containing only states/transitions which are in both FSAs
  - difference: contains all states/transitions which are in one but not the other FSA
  - complementation: FSA accepting all strings not accepted by the original one
  - reversal: FSA accepting all the reversed sequences accepted by the original one
- the results of these composition operators are FSAs again
- → algebra for computing with FSA

Finite state techniques

- Information extraction with FSAs
  - date and time expressions
  - named entity recognition

Finite state techniques

- Morphology with FSAs
  - concatenative morphology
    - inflection, derivation, compounding, clitization
    - prefixation, suffixation:

      `(re-)?emerg(e|es|ed|ing|er)`
      `(re)?load(s?|ed|ing|er)`
      `(re)?toss(es?|ed|ing|er)`
      `compl(y|ies|ied|ying|yer)`
      `enjoy(s?|ed|ing|er)`
    - linguistically unsatisfactory
  - non-concatenative morphology: reduplication, root-pattern phenomenon

Finite state techniques

- finite state transducers
  - transitions are labelled with pairs of symbols
  - sequences on different representation levels can be translatetd into each other
  - declarative formalism: translation can be in both directions
  - morphological processes can be separated from phonological ones

Finite state techniques

- two representational levels
  - lexical representation (concatenation of morphs)

    `emergeS`
    `tossS`
    `loadS`
    `complyS`
    `enjoyS`
  - phonological mapping (transformation to surface form)

`S` → `s+ / [^ys] _ .`	emerges, loads
`S` → `(es)+ / s _ .`	tosses
`yS` → `(ies|y) / [^ao] _ .`	complies
`yS` → `(ys|y) / [ao] _ .`	enjoys
  - similar models for other suffixes/prefixes

Finite state techniques

- FSTs can be non-deterministic: one input symbol can translate into alternative output symbols
- search required → expensive
- transformation of non-deterministic FSAs to deterministic ones?
  - only for special cases possible

Finite state techniques

- composition of FSTs
  - disjunction/union
  - inversion: exchange input and output
  - composition: cascading FSTs
  - intersection: only for ϵ-free FSTs (input and output has the same length)
- cascaded FSTs: multiple representation levels
- input string may also contain morpho-syntactic features (3sg, pl, ...)
- transformed to an intermediate representation
- phonologically spelled out

Finite state techniques

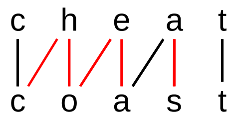
- root-pattern-phenomena

Finite state techniques

- limitations of finite state techniques
  - no languages with infinitely deeply nested brackets: $a^n b^n$
  - only segmentation of strings; no structural description can be generated
- advantages of finite state techniques
  - simple
  - formally well understood
  - efficient for typical problems of language processing
  - declarative (reverseable)

String-to-string matching

- measure for string similarity: minimum edit distance, Levenshtein-metric
- edit operations: substitution, insertion and deletion of symbols
- applications: spelling error correction, evaluation of word recognition results
- combines two tasks: alignment and error counting
- alignment: pairwise, order preserving mapping between the elements of the two strings
- alternative alignments with same distance possible



String-to-string matching

- string edit distance is a non-deterministic, recursive function

$$d(x_{0:0}, y_{0:0}) = 0$$

$$d(x_{1:m}, y_{1:n}) = \min \begin{cases} d(x_{2:m}, y_{2:n}) + c(x_1, y_1) \\ d(x_{1:m}, y_{2:n}) + c(\epsilon, y_1) \\ d(x_{2:m}, y_{1:n}) + c(x_1, \epsilon) \end{cases}$$

- Levenshtein metric: uniform cost function $c(.,.)$

String-to-string matching

- finding the minimum distance is an optimization problem
  $\rightarrow$ dynamic programming
- The locally optimal path to a state will be part of the global optimum if that state is part of the global optimum.
- all pairs of alignments need to be checked
- inverse formulation of the scoring function

$$d(x_{0:0}, y_{0:0}) = 0$$

$$d(x_{1:m}, y_{1:n}) = \min \begin{cases} d(x_{1:m-1}, y_{1:n-1}) + c(x_m, y_n) \\ d(x_{1:m}, y_{1:n-1}) + c(\epsilon, y_n) \\ d(x_{1:m-1}, y_{1:n}) + c(x_m, \epsilon) \end{cases}$$

String-to-string matching

- 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

		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

global distances

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

String-to-string matching

- string-to-string matching with Levenshtein metric is quite similar to searching a non-deterministic FSA
  - the search space is dynamically generated from one of the two strings
  - the other string is identified in the search space
- additional functionality
  - the number of ”error” transitions is counted
  - the minimum is selected

String-to-string matching

- limitation of the Levenshtein metric
  - uniform cost assignment
- but sometimes different costs for different error types desirable (keyboard layout, phonetic confusion)
  - consequence: alternative error sequences lead to different similarity values (SI vs. IS, SD vs DS)
- sometimes even special error types required: e.g. transposition of neighboring characters

Speech recognition 1: DTW

- Signal processing
- Dynamic time warping

Signal processing

- digitized speech signal is a sequence of numerical values (time domain)
- assumption: most relevant information about phones is in the frequency domain
- transformation becomes necessary
- spectral transformations are only defined for infinite (stationary) signals
- but speech signal is a highly dynamic process
- windowing: transforming short segments of the signal
- transformed signal is a sequence of feature vectors

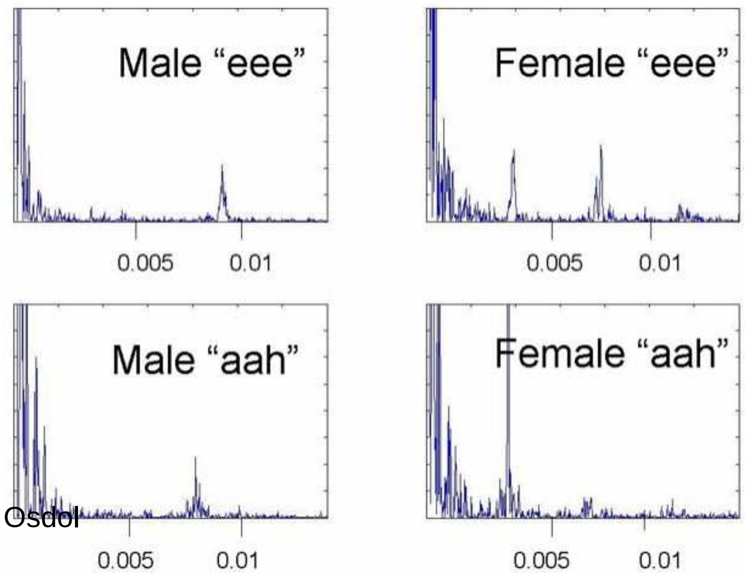
Signal processing

- Cepstral-coefficients
  - speech signal is convolution of the glottal exitation and the vocal tract shape
  - phone distictions are only depending on dynamics of the vocal tract
  - convolution is multiplication of the spectra
  - multiplication is the addition of the logarithms

$$C(m) = \mathcal{F}^{-1}(\hat{X}(k)) = \mathcal{F}^{-1}(\log(\mathcal{F}(x(n))))$$

Signal processing

- liftering: separation of the transfer function (spectral envelope) from the excitation signal



Brian van Osdol

Dynamic time warping

- simplest case of speech recognition: isolated words
- simplest method: dynamic time warping (DTW)
- first success story of speech recognition
- DTW is an instance based classifier:
  - compares the input signal to a list of stored pattern pronunciations
  - chooses the class of the sample which is closest to the input sequence
  - usually several sample sequences per word recorded

Dynamic time warping

- nearest-neighbor classifier

$$k(x[1{:}M]) = k(x_i[1{:}N_i])$$

$$\text{with } i = \arg\min_i d(x[1{:}M], x_i[1{:}N_i])$$

- two tasks:
  - alignment and distance measuring

Dynamic time warping

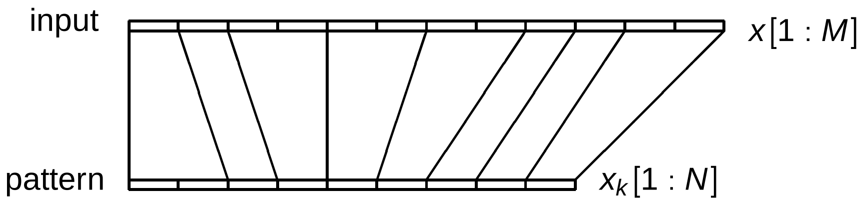
- distance of a pair of feature vectors: e.g. Euclidean metric

$$d(\vec{x},\vec{y}) = \sum_{i=1}^{I} (x_i - y_i)^2$$

- distance of two sequences of feature vectors: sum of the pairwise distance
- but length of spoken words varies
  - two instances of one and the same word are usually of different length
  - need to be squeezed or stretched to become comparable
- but dynamic variation is different for different phones
  - consonants are more stable than vowels

Dynamic time warping

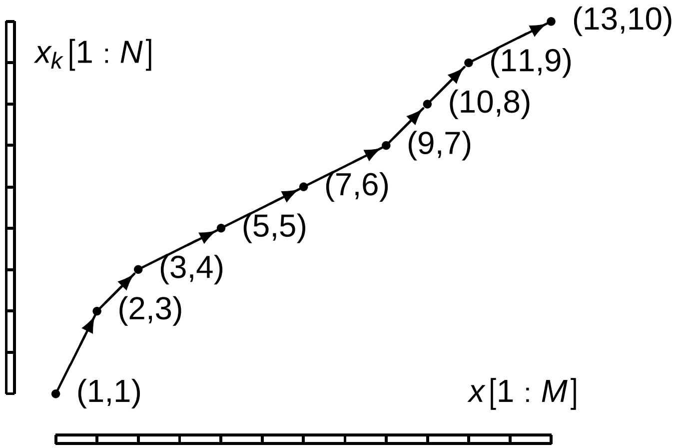
- non-linear time warping required



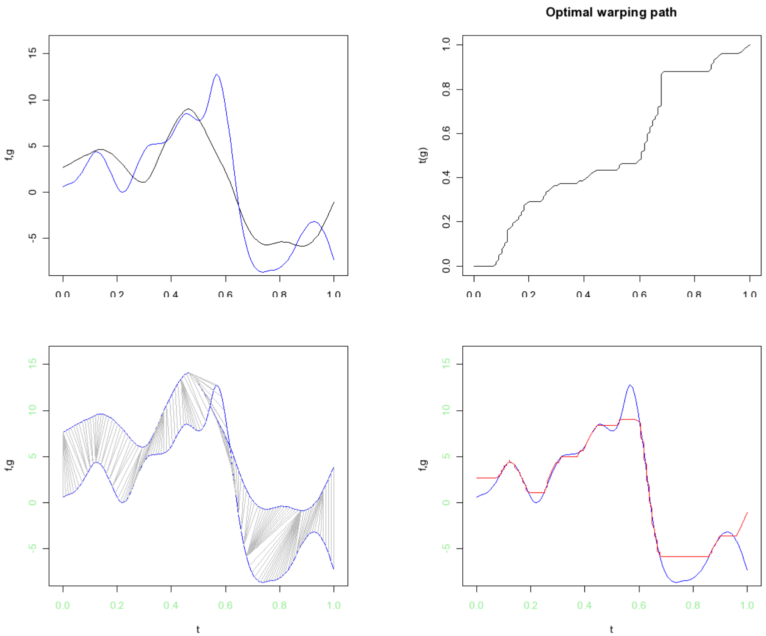
Dynamic time warping

- warping function

$$V = v_1 \dots v_I \quad \text{with} \quad v_i = (m_i, n_i)$$
$$d(v_i) = d(x[m_i], x_k[n_i])$$



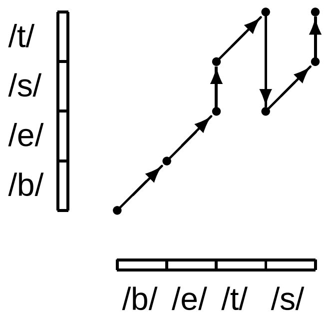
Dynamic time warping



TELESCA (2005)

Dynamic time warping

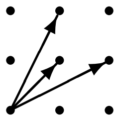
- not arbitrary warping functions are allowed
  - need to be monotonous



Dynamic time warping

- slope constraint for the warping function
- e.g. SAKOE-CHIBA with deletions

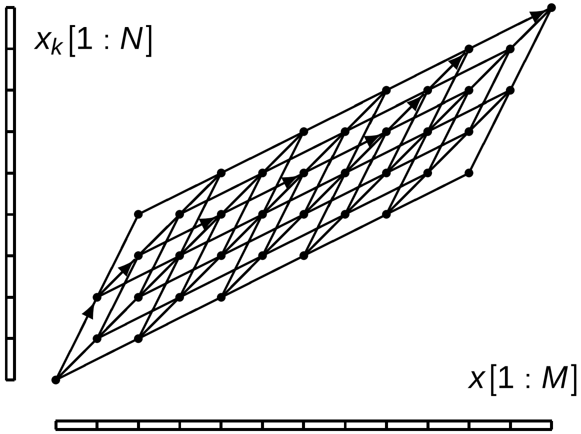
$$v_{i-1} = \begin{cases} (m_i - 1, n_i - 1) \\ (m_i - 2, n_i - 1) \\ (m_i - 1, n_i - 2) \end{cases}$$



- symmetrical slope constraint

Dynamic time warping

- trellis



Dynamic time warping

- distance between two vector sequences

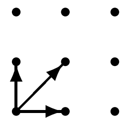
$$d(x[1{:}M], x_k[1{:}N]) = \min_{\forall V} \sum_{i=1}^{I} d(v_i)$$

V: warping functions

Dynamic time warping

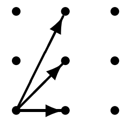
- alternative slope constraints
  - SAKOE-CHIBA without deletions

$$v_{i-1} = \begin{cases} (m_i - 1, n_i - 1) \\ (m_i, n_i - 1) \\ (m_i - 1, n_i) \end{cases}$$



  - ITAKURA (asymmetric)

$$v_{i-1} = \begin{cases} (m_i - 1, n_i) \\ (m_i - 1, n_i - 1) \\ (m_i - 1, n_i - 2) \end{cases}$$



    - requires additional global constraints
    - advantage: time synchroneous processing

Dynamic time warping

- algorithmic realisation: dynamic programming
  - search space is a graph defined by alternative alignment variants
  - search space is limited by the slope constraint
  - transitions are weighted (feature vector distance at the nodes)
  - task: finding the optimum path in the graph

Dynamic time warping

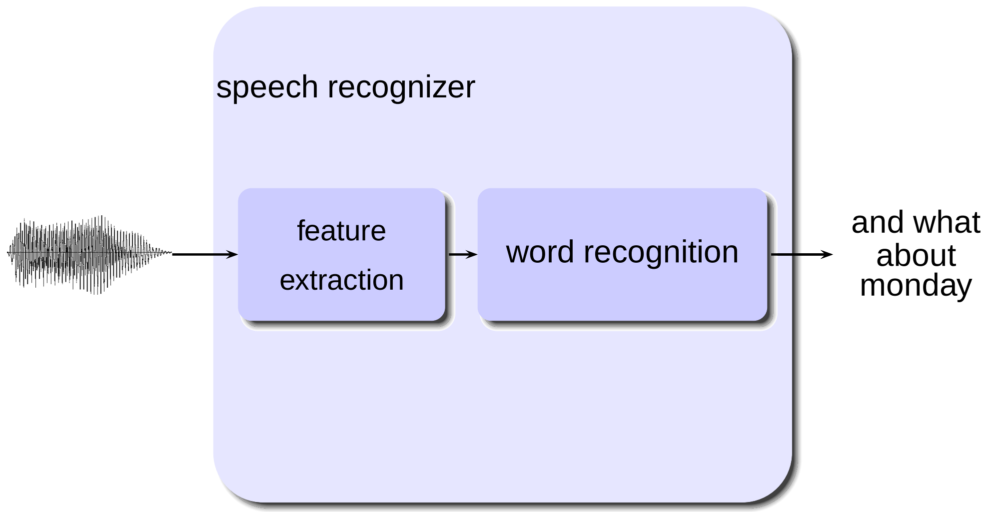
- redefining the global optimization problem in terms of local optimality decisions
- for ITAKURA constraint:

$$d(x[1{:}i], x_k[1{:}j])$$
$$= \min \left\{ \begin{array}{l} d(x[1{:}i-1], x_k[1{:}j]) \\ d(x[1{:}i-1], x_k[1{:}j-1]) \\ d(x[1{:}i-1], x_k[1{:}j-2]) \end{array} \right\} + d(x[i], x_k[j])$$

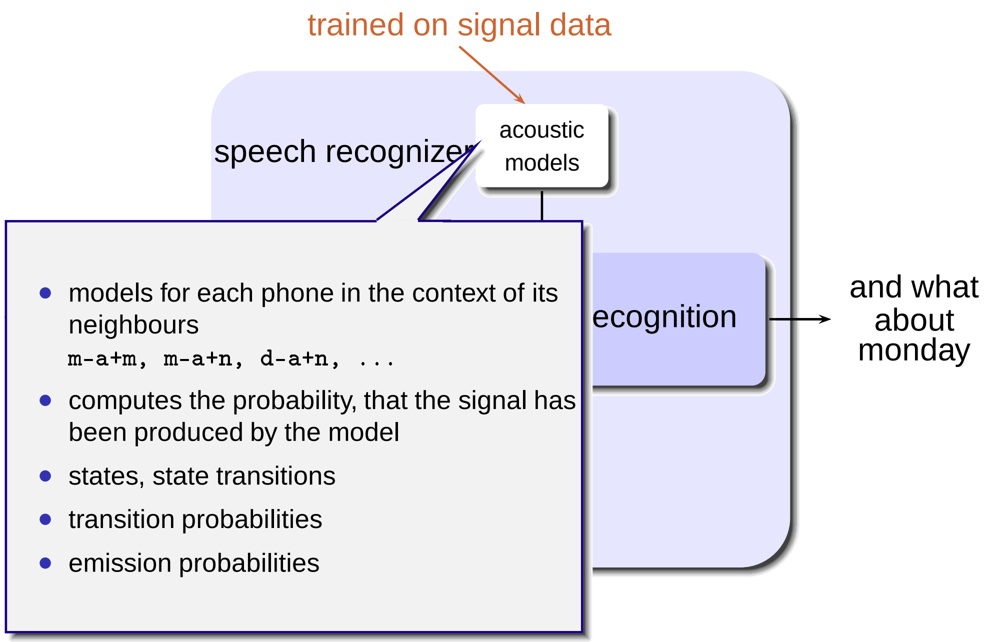
Dynamic time warping

- advantages:
  - simple training
  - simple recognition
- drawbacks:
  - highly speaker dependent

Speech recognition 2: HMM

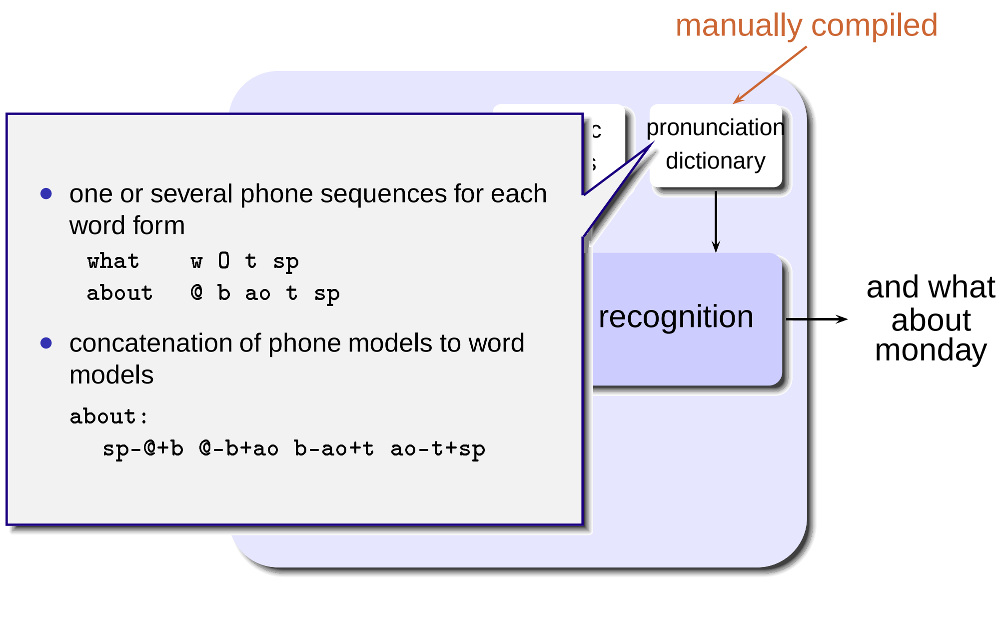


Speech recognition 2: HMM



- models for each phone in the context of its neighbours
  `m-a+m, m-a+n, d-a+n, ...`
- computes the probability, that the signal has been produced by the model
- states, state transitions
- transition probabilities
- emission probabilities

Speech recognition 2: HMM



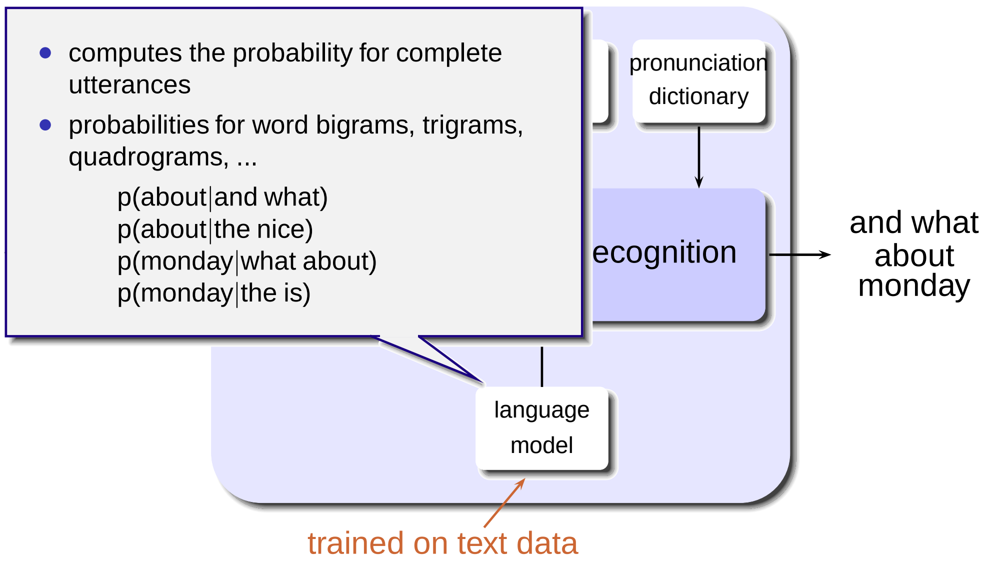
- one or several phone sequences for each word form

```
what    w O t sp
about   @ b ao t sp
```

- concatenation of phone models to word models

```
about:
  sp-@+b @-b+ao b-ao+t ao-t+sp
```

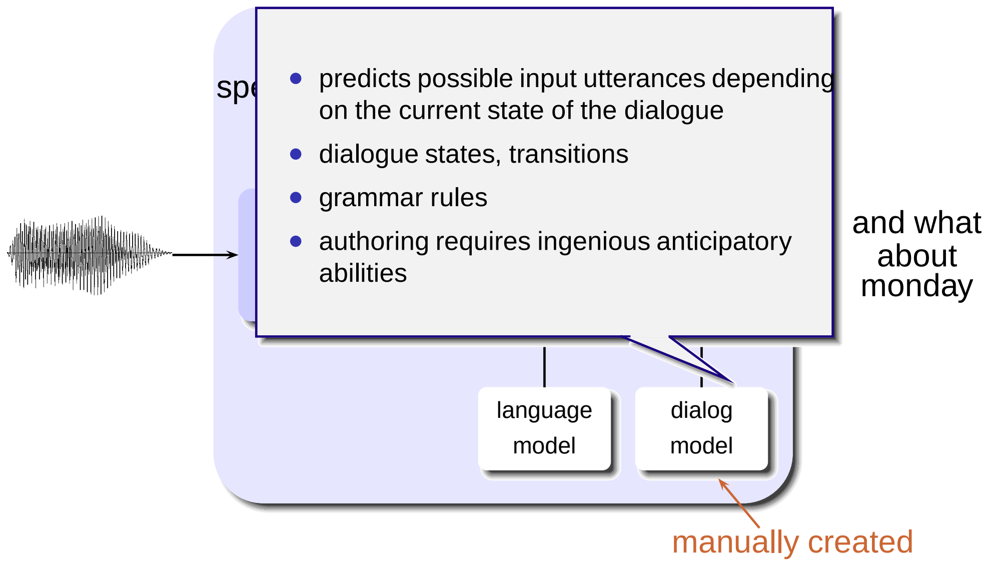

Speech recognition 2: HMM



- computes the probability for complete utterances
- probabilities for word bigrams, trigrams, quadrograms, ...

  p(about|and what)
  p(about|the nice)
  p(monday|what about)
  p(monday|the is)

Speech recognition 2: HMM



- predicts possible input utterances depending on the current state of the dialogue
- dialogue states, transitions
- grammar rules
- authoring requires ingenious anticipatory abilities

Speech recognition 2: HMM

- acoustic modelling
- word recognition
- HMM training
- stochastic language modelling
- dialog modelling

Acoustic modelling

- the problem: segment boundaries are not reliably detectable prior to the phone classification
- the solution: classify phone sequences
- formal foundation: Markov models

Acoustic modelling

- Bayesian decision theory (error optimal!)

$$\begin{aligned} c(\vec{x}) &= \arg\max_i P(c_i|\vec{x}) \\ &= \arg\max_i \frac{P(c_i) \cdot P(\vec{x}|c_i)}{P(\vec{x})} \\ &= \arg\max_i P(c_i) \cdot P(\vec{x}|c_i) \end{aligned}$$

- atomic observations $\mapsto$ atomic class assignments
- isolated word recognition:
  sequential observations $\mapsto$ atomic class decision

$$c(x[1:n]) = \arg\max_i P(c_i) \cdot P(x[1:n]|c_i)$$

Acoustic modelling

- continuous speech recognition:
  sequential observations $\mapsto$ sequences of class decisions

$$c(x[1:n]) = \arg\max_{m, c[1:m]} P(c[1:m]) \cdot P(x[1:n]|c[1:m])$$

$\rightarrow$ Markov models

Acoustic modelling

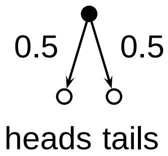
$$c(x[1:n]) = \arg\max_{m,c[1:m]} P(c[1:m]) \cdot P(x[1:n] | c[1:m])$$

language model

acoustic model

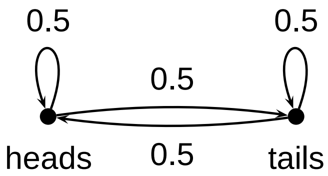
Acoustic modelling

- to provide the necessary flexibility for training
  → hidden Markov models
  - doubly stochastic process
  - states which change stochastically
  - observations which are emitted from states stochastically
- the same observation distributions can be modelled by quite different parameter settings
- example: coin
- emission probability only

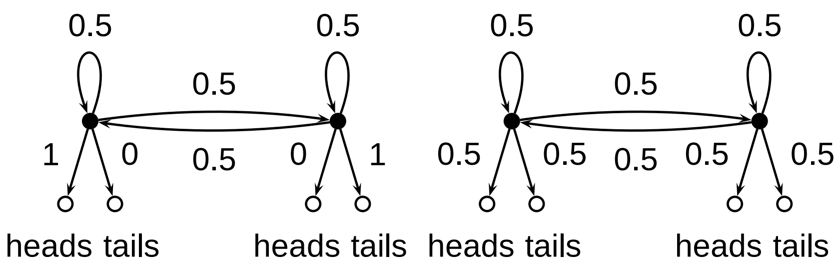


Acoustic modelling

- transition proabilities only (1st order Markov model)

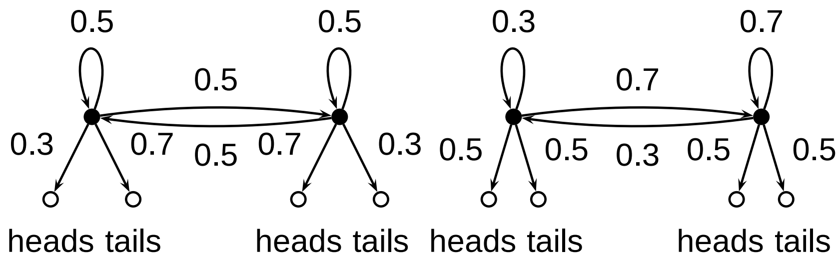


- Hidden Markov Models for the observation



Acoustic modelling

- alternative HMMs for the same observation



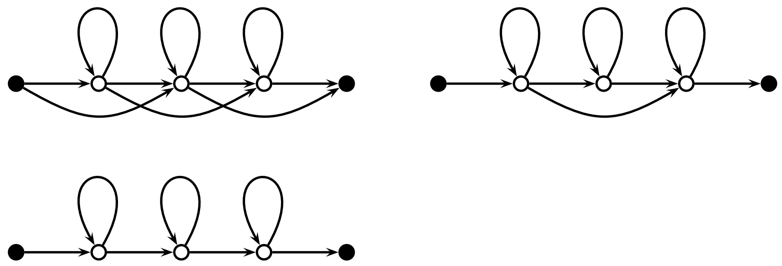
- even more possibilities for biased coins or coins with more than two sides

Acoustic modelling

- phone recognition: identifying differently biased coins
  - train different HMMs for the different coins: adjust the probabilities so that they predict a training sequence of observations with maximum probability
  - determine the model which predicts the observed (test) sequence of feature verctors with the highest probability

Acoustic modelling

- model topologies for phones (only transitions depicted)



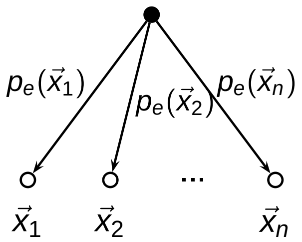
the more data available $\rightarrow$ the more sophisticated models can be trained

Acoustic modelling

- monophone models do not capture coarticulatory variation $\rightarrow$ triphone models
- triphone: context sensitive phone model
  - increases the number of models to be trained
  - decreases the amount of training data available per model
  - context clustering to share models across contexts
- special case: cross word triphones (expensive to be used)

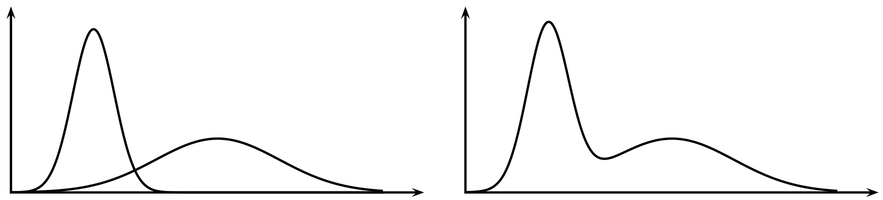
Acoustic modelling

- modelling of emission probabilities
- discrete models: quantized feature vectors
  - local regions of the feature space are represented by a prototype vector
  - usually 1024 or 2048 prototype vectors



Acoustic modelling

- continuous models: probability distributions for feature vectors
- usually multidimensional Gaussian mixtures
- extension to mixture models



$$p(x|s_i) = \sum_{m=1}^{M} c_m \, \mathcal{N}[x, \mu_m, \Sigma_m] \qquad \mathcal{N}[x, \mu, \sigma] = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- number of mixtures is chosen according to the available training material

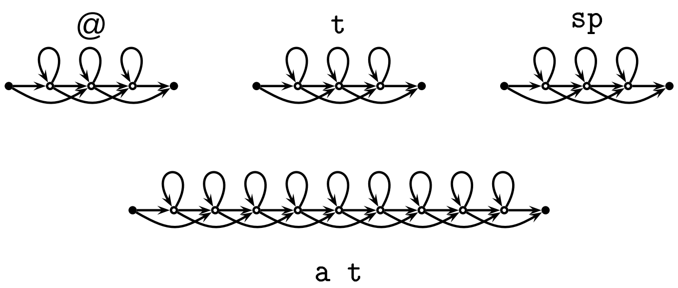
Acoustic modelling

- dealing with data sparseness
  - sharing of mixture components: semi-continuous models
  - sharing of mixture distributions: tying of states
  - parameter reduction: restriction to diagonal covariance matrices
- speaker adaptation techniques
  - retraining with speaker specific data
  - vocal length estimation $\rightarrow$ global transform of the feature space
  - ...

Word recognition

- concatenate the phone models to word models based on the information from the pronunciation dictionary

`at        @ t sp`



- apply all the word models in parallel
- choose the one which fits the data best

Word recognition

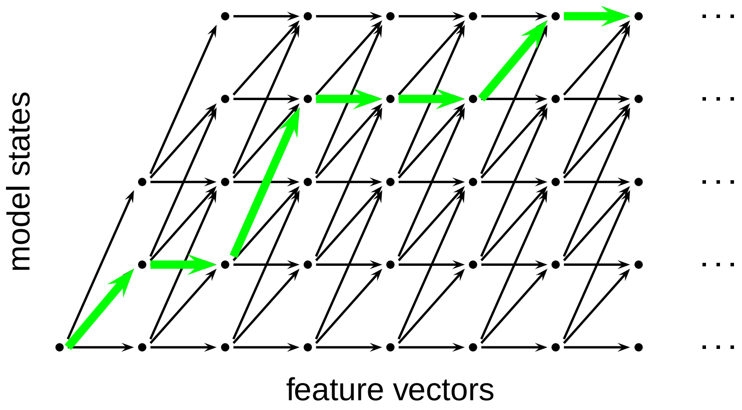
- recognition of continuous speech: Viterbi search
- find the *path* through the model which generates the signal observation with the highest probability

$$p(x[1:n]|s_i) = \max_{s_i = succ(s_j)} p(x[1:n-1]|s_j) \cdot p_t(s_i|s_j) \cdot p_e(s_i|x(n))$$

- recursive decomposition: special case of a dynamic programming algorithm
- linear with the length of the input

Word recognition

- model topology unfolds the search space into a tree with a limited branching factor
- model state and time indicees are used to recombine search paths
- maximum decision rule facilitates unique path selection



HMM training

- concatenate the phone models according to the annotation of the training data into a single model
- Baum-Welch reestimation
  - iterative refinement of an initial value assignment
  - special case of an expectation maximization (EM) algorithm
  - gradient ascend: cannot guarantee to find the optimum model
- word level annotations are sufficient
- no prior segmentation of the training material necessary

Stochastic language modelling

- idea: mimick the expectation driven nature of human speech comprehension

What's next in an utterance?

- stochastic language models $\rightarrow$ free text applications
- grammar-based language models $\rightarrow$ dialog modelling
- combinations

Stochastic language modelling

- n-grams: $p(w_i|w_{i-1})$ $p(w_i|w_{i-2}w_{i-1})$
- trained on huge amounts of text
- most probabilities are zero: n-gram has been never observed, but could occur in principle
- backoff: if a probability is zero, approximate it by means of the next less complex one
  - trigram $\rightarrow$ bigram
  - bigram $\rightarrow$ unigram

Stochastic language modelling

- perplexity: ”ambiguity” of a stochastic source

$$Q(S) = 2^{H(S)}$$

- $H(S)$ entropy of a source S, which emits symbols $w \in W$

$$H(S) = -\sum_{w} p(w) \log_2 p(w)$$

- perplexity is used to decribe the restrictive power of a probabilistic language model and/or the difficulty of a recognition task
- test set perplexity

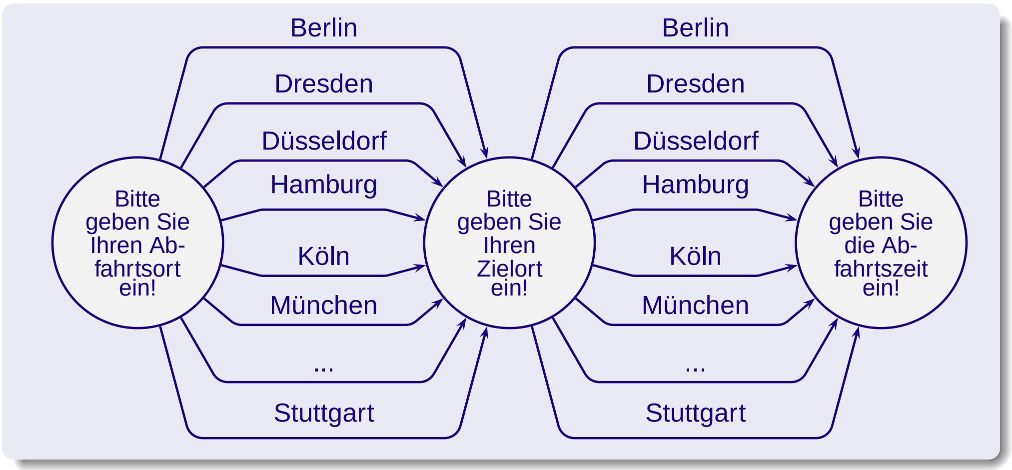
$$Q(T) = 2^{H(T)} = p(w[1:n])^{-\frac{1}{n}}$$

Dialog modelling

- based on dialog states: What’s next in a dialogue?
- reducing the number of currently active lexical items
  - to increase recognition accuracy
  - e.g by avoiding confusables
- simplifying semantic interpretation
  - context-based disambiguation between alternative interpretation possibilities
  - e.g. number $\rightarrow$ price, time, date, account number, ...

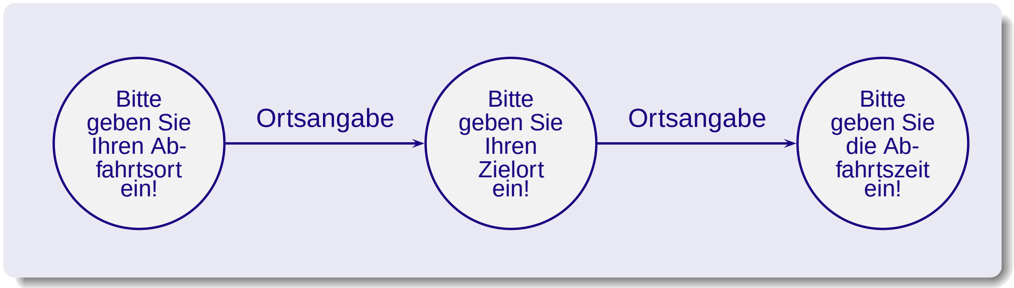
Dialog modelling

- dialog states: input request (prompt)
- transitions between states: possible user input



Dialog modelling

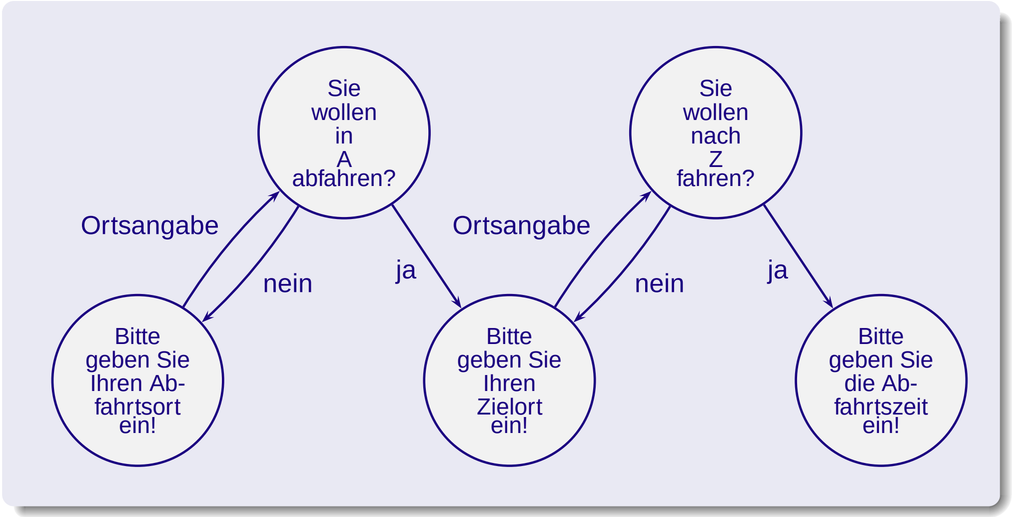
- recycling of partial networks



- set of admissible utterances can also be specified by means of generative grammars

Dialog modelling

- confirmation dialogs: compensating recognition uncertainty



Dialog modelling

- finite state automata are very rigid
- relaxing the constraints
  - partial match
  - barge in
- flexible mechanisms for dynamically modifying system prompts
  - less monotonous human computer interaction
  - simple forms of user adaptation

POS-Tagging

- lexical categories
- constraint-based tagger
- stochastic tagger
- transformation-based tagger
- applications

Lexical categories

- phonological evidence: explanation of systematic pronunciation variants

  *We need to in***crease** *productivity.*
  We need an **in***crease in productivity.*
  *Why do you tor***ment** *me?*
  Why do you leave me in **tor***ment?*
  *We might trans***fer** *him to another club.*
  He's asked for a **trans***fer.*

- semantic evidence: explanation of structural ambiguities

  Mistrust wounds.

  semantic properties itself are irrelevant

Lexical categories

- morphological evidence
  - different inflectional patterns for verbs, nouns, and adjectives
    - but: irregular inflection; e.g. strong verbs, *to be*
  - different word formation pattern
    - deverbalisation: -tion
    - denominalisation: -al

Lexical categories

- syntactic evidence: distributional classes
  - nouns

    Linguistics can be a pain in the neck.
    John can be a pain in the neck.
    Girls can be a pain in the neck.
    Television can be a pain in the neck.
    ** Went can be a pain in the neck.*
    ** For can be a pain in the neck.*
    ** Older can be a pain in the neck.*
    ** Conscientiously can be a pain in the neck.*
    ** The can be a pain in the neck.*

Lexical categories

- tagsets
  - inventories of categories for the annotation of corpora
  - sometimes even morpho-syntactic subcategories (plural, ...)
  - ”technical” tags
    - foreign words, symbols, interpunction, ...

Penn-Treebank	Marcus et al. (1993)	45
British National Corpus (C5)	Garside et al. (1997)	61
British National Corpus (C7)	Leech et al. (1994)	146
Tiger (STTS)	Schiller, Teufel (1995)	54
Prague Treebank	Hajic (1998)	3000/1000

Lexical categories

- Penn-Treebank (Marcus, Santorini, Marcinkiewicz 1993)

CC	Coordinating conjunction	*and,but,or, ...*
CD	Cardinal Number	*one, two, three, ...*
DT	Determiner	*a, the*
EX	Existential *there*	*there*
FW	Foreign Word	*a priori*
IN	Preposition or subord. conjunction	*of, in, by, ...*
JJ	Adjective	*big, green, ...*
JJR	Adjective, comparative	*bigger, worse*
JJS	Adjective, superlative	*lowest, best*
LS	List Item Marker	*1, 2, One, ...*
MD	Modal	*can, could, might, ...*
NN	Noun, singular or mass	*bed, money, ...*
NNP	Proper Noun, singular	*Mary, Seattle, GM, ...*
NNPS	Proper Noun, plural	*Koreas, Germanies, ...*
NNS	Noun, plural	*monsters, children, ...*

Lexical categories

- Penn-Treebank (2)

PDT	Predeterminer	*all, both, ... (of the)*
POS	Possessive Ending	*'s*
PRP	Personal Pronoun	*I, me, you, he, ...*
PRP$	Possessive Pronoun	*my, your, mine, ...*
RB	Adverb	*quite, very, quickly, ...*
RBR	Adverb, comparative	*faster, ...*
RBS	Adverb, superlative	*fastest, ...*
RP	Particle	*up, off, ...*
SYM	Symbol	+, %, & ...
TO	*to*	*to*
UH	Interjection	*uh, well, yes, my, ...*
VB	Verb, base form	*write, ...*
VBD	Verb, past tense	*wrote, ...*
VBG	Verb, gerund	*writing*
VBN	Verb, past participle	*written, ...*

Lexical categories

- Penn-Treebank (3)

VBP	Verb, non-3rd singular present	*write, ...*
VBZ	Verb, 3rd person singular present	*writes, ...*
WDT	Wh-determiner	e.g. *which, that*
WP	Wh-pronoun	e.g. *what, whom, ...*
WP$	Possessive wh-pronoun	*whose, ...*
WRB	Wh-adverb	e.g. *how, where, why*
$	Dollar sign	*$*
#	Pound sign	*#*
”	left quote	*”*
´´	right quote	*´´*
(	left parantheses	*(*
)	right parantheses	*)*
,	comma	*,*
.	sentence final punct.	*., !, ?*
:	mid-sentence punct.	*:, ;, –, ...*

Lexical categories

- Examples

  Book/NN/VB that/DT/WDT flight/NN ./.

  Book/VB that/DT flight/NN ./.

Constraint-based tagger

- ENGTWOL, Helsinki University (Voutilainen 1995)
- two-step approach
  - assignment of POS-hypotheses: morphological analyzer (two-level morphology)
  - selection of POS-hypotheses (constraint-based)
- lexicon with rich morpho-syntactic information

```
("<round>"
  ("round" <SVO><SV> V SUBJUNCTIVE VFIN (@+FMAINV))
  ("round" <SVO><SV> V IMP VFIN (@+FMAINV))
  ("round" <SVO><SV> V INF)
  ("round" <SVO><SV> V PRES -SG3 VFIN (@+FMAINV))
  ("round" PREP)
  ("round" N NOM SG)
  ("round" A ABS)
  ("round" ADV ADVL (@ADVL)))
```

Constraint-based tagger

- 35-45% of the tokens are ambiguous: 1.7-2.2 alternatives per word form
- hypothesis selection by means of constraints (1100)
  - linear sequence of morphological features
- example
  - input: *a reaction* ***to*** *the ringing of a bell*
  - dictionary entry:

```
("<to>"
  ("to" PREP)
  ("to" INFMARK> (@INFMARK>))
```


Constraint-based tagger

- example
  - constraint

```
("<to>" =0 (INFMARK>) (NOT 1 INF)
                      (NOT 1 ADV)
                      (NOT 1 QUOTE)
                      (NOT 1 EITHER)
                      (NOT 1 SENT-LIM))
```

Remove the infinitival reading if immediately to the right of *to* no infinitive, adverb, citation, *either, neither, both* or sentence delimiter can be found.

Constraint-based tagger

- quality measures
  - measurement on an annotated testset (“gold standard”)

$$\text{recall} = \frac{\text{retrieved correct categories}}{\text{actually correct categories}}$$

$$\text{precision} = \frac{\text{retrieved correct categories}}{\text{retrieved categories}}$$

- recall $<$ 100%: erroneous classifications
- recall $<$ precision: incomplete category assignment
- recall = precision: fully disambiguated output
  $\rightarrow$ accuracy
- recall $>$ precision: incomplete disambiguation

Constraint-based tagger

- ENGTWOL:
  - testset: 2167 word form token
  - recall: 99.77 %
  - precision: 95.94 %

  $\rightarrow$ incomplete disambiguation

Constraint-based tagger

- How good are the results?
  1. upper limit: How good is the annotation?
     - 96-97% agreement between annotators (MARCUS ET AL. 1993)
     - almost 100% agreement in case of negotiation (VOUTILAINEN 1995)
  2. lower limit: How good is the classifier?
     - baseline:
       e.g. most frequent tag (unigram probability)
     - example: $P(\text{NN}|\textit{race}) = 0.98$ $P(\text{VB}|\textit{race}) = 0.02$
     - 90-91% precision/recall (CHARNIAK ET AL. 1993)

Constraint-based tagger

- manual compilation of the constraint set
  - expensive
  - error prone
- alternative: machine learning components

Stochastic tagger

- noisy-channel model
  - mapping from word forms to tags is not deterministic
  - ”noise” of the channel depends on the context
  - model with memory: Markov model
  - memory is decribed by means of states
  - parameters of the model describe the probability of a state transition
    - transition probabilities: $P(s_i|s_1 \ldots s_{i-1})$
- hidden markov models
  - observations are not strictly coupled to the transitions
  - sequence of state transition influences the observation sequence only stochastically
    - emission probabilities: $P(o_i|s_1 \ldots s_{i-1})$

Stochastic tagger

- model topologies for HMM taggers
  - observations: word forms w_i
  - states: tags t_i
  - transition probabilities: $P(t_i|t_1 \ldots t_{i-1})$
  - emission probabilities: $P(w_i|t_1 \ldots t_{i-1})$

Stochastic tagger

- classification: computation of the most probable tag sequence

$$t_j[1,n] = \arg\max_{t[1,n]} P(t[1,n]|w[1,n])$$

- Bayes' Rule

$$t_j[1,n] = \arg\max_{t[1,n]} \frac{P(t[1,n]) \cdot P(w[1,n]|t[1,n])}{p(w[1,n])}$$

- probability of the word form sequence is constant for a given observation and therefore has no influence on the decision result

$$t_j[1,n] = \arg\max_{t[1,n]} P(t[1,n]) \cdot P(w[1,n]|t[1,n])$$

Stochastic tagger

- chain rule for probabilities

$$\begin{aligned} P(t[1,n]) \cdot P(w[1,n] \mid t[1,n]) \\ = \prod_{i=1}^{n} P(t_i \mid w_1 t_1 \ldots w_{i-1} t_{i-1}) \\ \cdot P(w_i \mid w_1 t_1 \ldots w_{i-1} t_{i-1} t_i) \end{aligned}$$

$$\begin{aligned} t_j[1,n] = \arg\max_{t[1,n]} \\ \prod_{i=1}^{n} P(t_i \mid w_1 t_1 \ldots w_{i-1} t_{i-1}) \\ \cdot P(w_i \mid w_1 t_1 \ldots w_{i-1} t_{i-1} t_i) \end{aligned}$$

Stochastic tagger

- 1st simplification: the word form only depends on the current tag

$$\begin{aligned} t_j[1,n] = \arg\max_{t[1,n]} \\ \prod_{i=1}^{n} P(t_i \mid w_1 t_1 \ldots w_{i-1} t_{i-1}) \cdot P(w_i \mid t_i) \end{aligned}$$

- 2nd simplification: the current tag depends only on its predecessors (not on the observations!)

$$t_j[1,n] = \arg\max_{t[1,n]} \prod_{i=1}^{n} P(t_i \mid t_1 \ldots t_{i-1}) \cdot P(w_i \mid t_i)$$

Stochastic tagger

- 3rd simplification: the current tag depends only on its two predecessors
  - limited memory (Markov assumption): Trigram-Modell

$$t_j[1,n] = \arg\max_{t[1,n]} \prod_{i=1}^{n} P(t_i \mid t_{i-1} t_{i-2}) \cdot P(w_i \mid t_i)$$

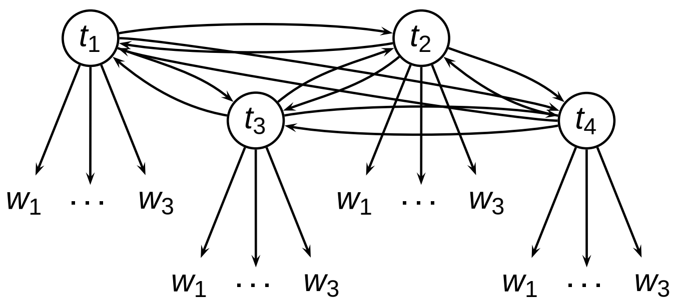
$\rightarrow$ 2nd order Markov process

Stochastic tagger

- further simplification leads to a bigram model
  - stochastic dependencies are limited to the immediate predecessor

$$t_j[1,n] = \arg\max_{t[1,n]} \prod_{i=1}^{n} P(t_i \mid t_{i-1}) \cdot P(w_i \mid t_i)$$

$\rightarrow$ 1st order Markov process



Stochastic tagger

- computation of the most likely tag sequence by dynamic programming (Viterbi, Bellmann-Ford)

$$\alpha_n = \max_{t[1,n]} \prod_{i=1}^{n} P(t_i \mid t_{i-1}) \cdot P(w_i \mid t_i)$$

$$\alpha_n = \max_{t_{n-1}} \; P(t_n \mid t_{n-1}) \cdot P(w_n \mid t_n) \cdot \alpha_{n-1}$$

- sometimes even local decision taken (greedy search)
- scores can be interpreted as confidence values

Stochastic tagger

- training: estimation of the probabilities
  - transition probabilities

$$P(t_i \mid t_{i-2}t_{i-1}) = \frac{c(t_{i-2}t_{i-1}t_i)}{c(t_{i-2}t_{i-1})}$$

  - emission probabilities

$$P(w_i \mid t_i) = \frac{c(w_i, t_i)}{c(t_i)}$$

Stochastic tagger

- unseen transition probabilities
  - backoff: using bigram or unigram probabilities

$$P(t_i|t_{i-2}t_{i-1}) = \begin{cases} P(t_i|t_{i-2}t_{i-1}) & \text{if} \quad c(t_{i-2}t_{i-1}t_i) > 0 \\ P(t_i|t_{i-1}) & \text{if} \quad c(t_{i-2}t_{i-1}t_i) = 0 \\ & \quad \text{and} \;\; c(t_{i-1}t_i) > 0 \\ P(t_i) & \text{else} \end{cases}$$

Stochastic tagger

- unseen transition probabilities
  - interpolation: merging of the trigram with the bigram and unigram probabilities

$$P(t_i|t_{i-2}t_{i-1}) = \lambda_1 P(t_i|t_{i-2}t_{i-1}) + \lambda_2 P(t_i|t_{i-1}) + \lambda_3 P(t_i)$$

    - λ_1, λ_2 and λ_3 are context dependent parameters
    - global constraint: $\lambda_1 + \lambda_2 + \lambda_3 = 1$
    - are trained on a separate data set (development set)

Stochastic tagger

- unseen word forms
  - estimation of the tag probability based on ”suffixes” (and if possible also on ”prefixes”)
- unseen POS assignment
  - smoothing
  - redistribution of probability mass from the seen to the unseen events (discounting)
  - e.g. WITTEN-BELL discounting (WITTEN-BELL 1991)
    - probability mass of the observation seen once is distributed to all the unseen events

Stochastic tagger

- example: TnT (BRANTS 2000)

corpus	share of unseen word forms	accuracy known word forms	accuracy unknown word forms	accuracy overall
PennTB (engl.)	2.9%	97.0%	85.5%	96.7%
Negra (dt.)	11.9%	97.7%	89%	96.7%
Heise (dt.)*)				92.3%

*) training data $\neq$ test data

- maximum entropy tagger (RATNAPARKHI 1996): 96.6%

Transformation-based tagger

- ides: stepwise correction of wrong intermediate results (BRILL 1995)
  - context-sensitive rules, e.g.
    Change NN to VB when the previous tag is TO
- rules are trained on a corpus
  1. initialisation: choose the tag sequence with the highest unigram probability
  2. compare the results with the gold standard
  3. generate a rule, which removes most errors
  4. run the tagger again and continue with 2.
- stop if no further improvement can be achieved

Transformation-based tagger

- rule generation driven by templates
  - change tag *a* to tag *b* if ...
    ...the preceding/following word is tagged *z*.
    ...the word two before/after is tagged *z*.
    ...one of the two preceding/following words is tagged *z*.
    ...one of the three preceding/following words is tagged *z*.
    ...the preceding word is tagged *z* and the following word is tagged *w*.
    ...the preceding/following word is tagged *z* and the word two before/after is tagged *w*.

Transformation-based tagger

- results of training: ordered list of transformation rules

from	to	condition	example
NN	VB	previous tag is TO	to/TO race/NN → VB
VBP	VB	one of the 3 previous tags is MD	might/MD vanish/VBP → VB
NN	VB	one of the 2 previous tags is MD	might/MD not reply/NN → VB
VB	NN	one of the 2 previous tags is DT	
VBD	VBN	one of the 3 previous tags is VBZ	

Transformation-based tagger

- 97.0% accuracy, if only the first 200 rules are used
- 96.8% accuracy with the first 100 rules
- quality of a HMM tagger on the same data (96.7%) is achieved with 82 rules
- extremly expensive training
  $\approx 10^6$ times of a HMM tagger

Applications

- word stress in speech synthesis
  'content/NN con'tent/JJ
  'object/NN ob'ject/VB
  'discount/NN dis'count/VB
- computation of the stem (e.g. document retrieval)
- class based language models for speech recognition
- "shallow" analysis, e.g. for information extraction
- preprocessing for parsing data, especially in connection with data driven parsers

Part 3: Dealing with structures

- Dependency parsing
- Phrase-structure parsing
- Unification-based grammars
- Constraint-based models (HPSG)

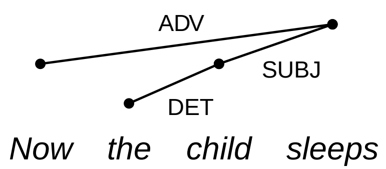
Dependency parsing

- Dependency structures
- Dependency parsing as constraint satisfaction
- Structure-based dependency parsing
- History-based dependency parsing
- Parser combination

Dependency structures

- labelled word-to-word dependencies

$$S \subset W \times W \times L$$



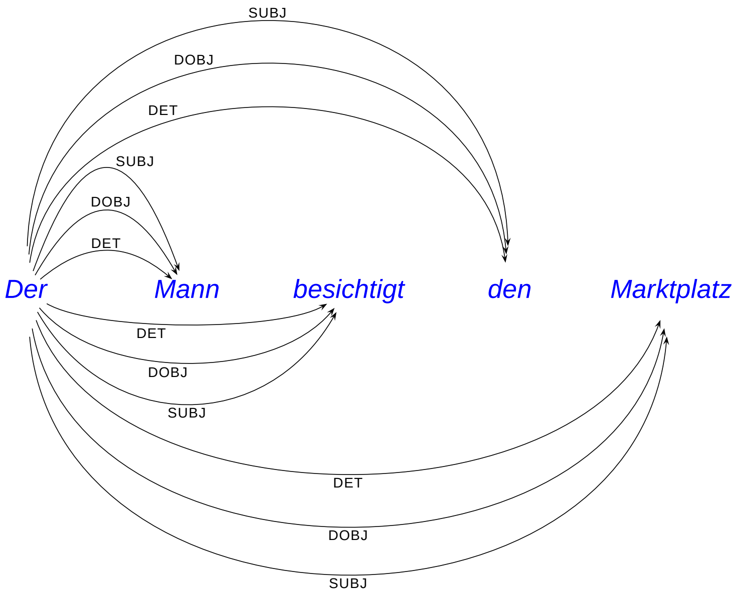
- distributional tests
  - attachment: deletion test
  - labelling: substitution test

Dependency structures

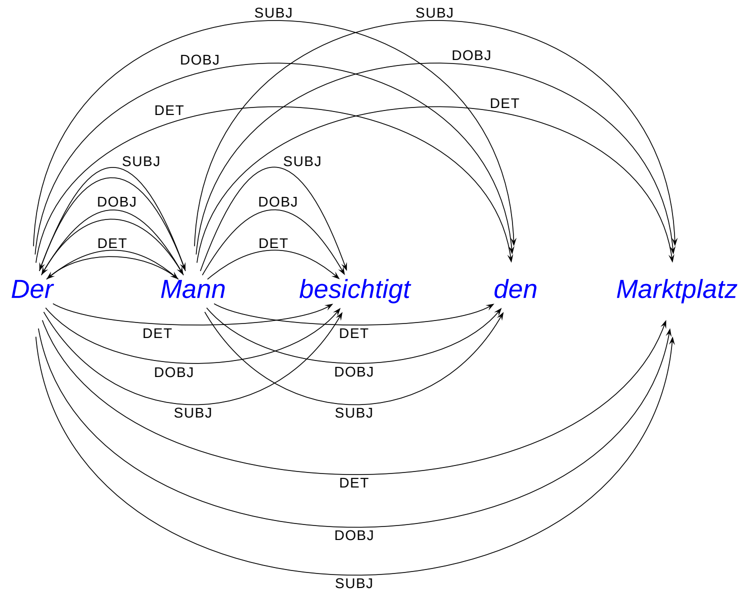
- highly regular search space

root/nil	root/nil	root/nil	root/nil	root/nil
det/2	det/1	det/1	det/1	det/1
det/3	det/3	det/2	det/2	det/2
det/4	det/4	det/4	det/3	det/3
det/5	det/5	det/5	det/5	det/4
subj/2	subj/1	subj/1	subj/1	subj/1
subj/3	subj/3	subj/2	subj/2	subj/2
subj/4	subj/4	subj/4	subj/3	subj/3
subj/5	subj/5	subj/5	subj/5	subj/4
dobj/2	dobj/1	dobj/1	dobj/1	dobj/1
dobj/3	dobj/3	dobj/2	dobj/2	dobj/2
dobj/4	dobj/4	dobj/4	dobj/3	dobj/3
dobj/5	dobj/5	dobj/5	dobj/5	dobj/4
Der	*Mann*	*besichtigt*	*den*	*Marktplatz*
1	2	3	4	5

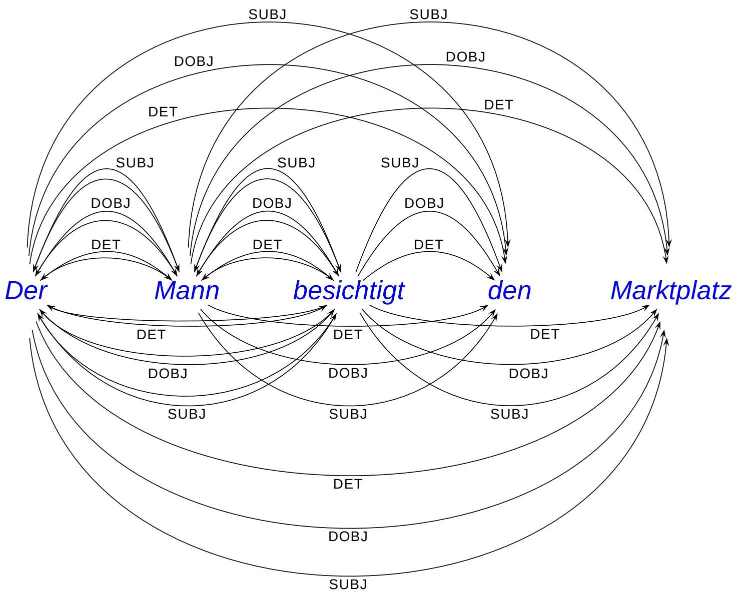
Hypothesis Space



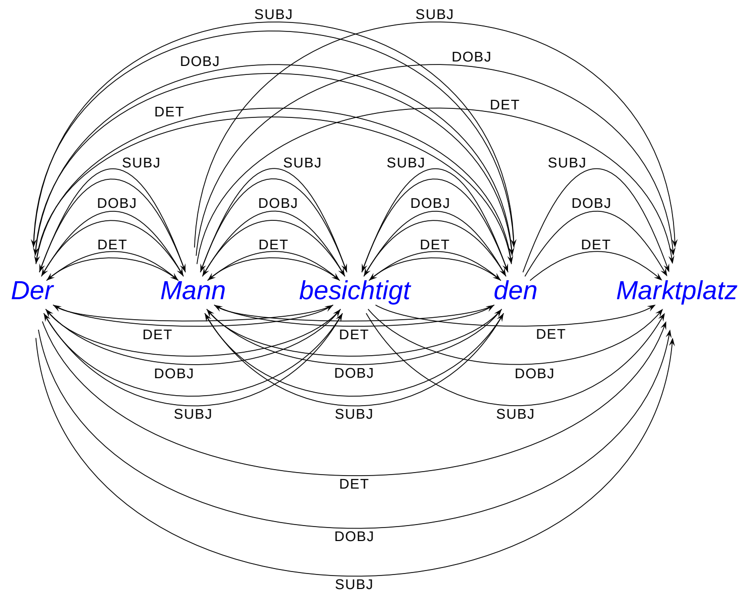
Hypothesis Space



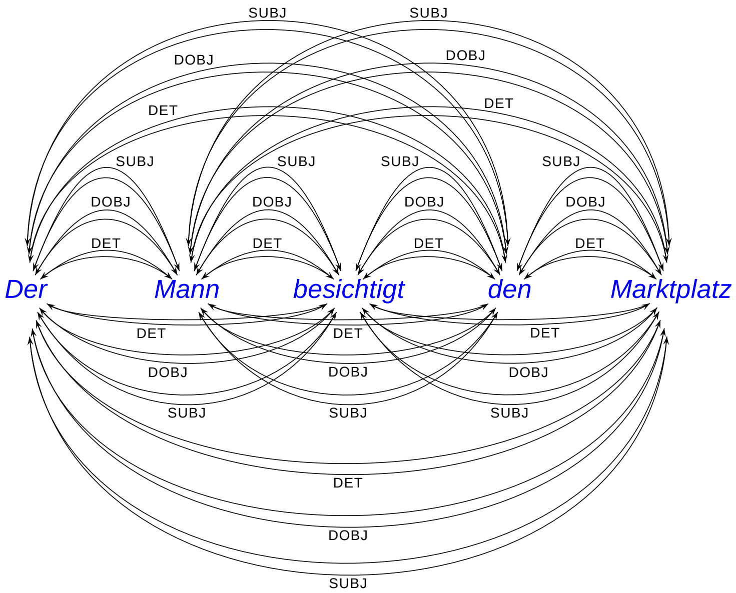
Hypothesis Space



Hypothesis Space



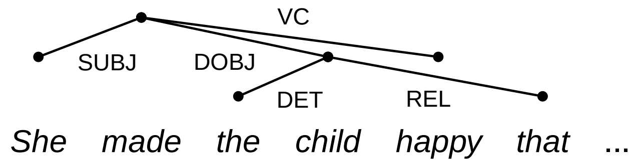
Hypothesis Space



Root attachments are not depicted.

Dependency structures

- source of complexity problems: non-projective trees



Dependency Modeling

- advantages (COVINGTON 2001, NIVRE 2005)
  - straightforward mapping of head-modifier relationships to arguments in a semantic representation
  - parsing relates existing nodes to each other
    - no need to postulate additional ones
  - word-to-word attachment is a more fine-grained relationship compared to phrase structures
    - modelling constraints on partial ”constituents”
    - factoring out dominance and linear order
    - well suited for incremental processing
  - non-projectivities can be treated appropriately
    - discontinuous constructions are not a problem

Dependency parsing as constraint satisfaction

- Constraint Grammar KARLSSON 1995
  - attaching possibly underspecified dependency relations to the word forms of an utterances

`@+FMAINV`	finite verb of a sentence
`@SUBJ`	grammatical subject
`@OBJ`	direct Object
`@DN>`	determiner modifying a noun to the right
`@NN>`	noun modifying a noun to the right

Dependency parsing as constraint satisfaction

- typical CS problem:
  - constraints: conditions on the (mutual) compatibility of dependency labels
  - indirect definition of well-formedness: everything which does not violate a constraint explicitly is acceptable
- strong similarity to tagging procedures

Dependency parsing as constraint satisfaction

- two important prerequisites for robust behaviour
  - inherent fail-soft property: the last remaining category is never removed even if it violates a constraint
  - possible structures and well-formedness conditions are fully decoupled: missing grammar rules do not lead to parse failures
- complete disambiguation cannot always be achieved

Bill	*saw*	*the*	*little*	*dog*	*in*	*the*	*park*
@SUBJ	@+FMAINV	@DN>	@AN>	@OBJ	@<NOM	@DN>	@<P
					@<ADVL		

Dependency parsing as constraint satisfaction

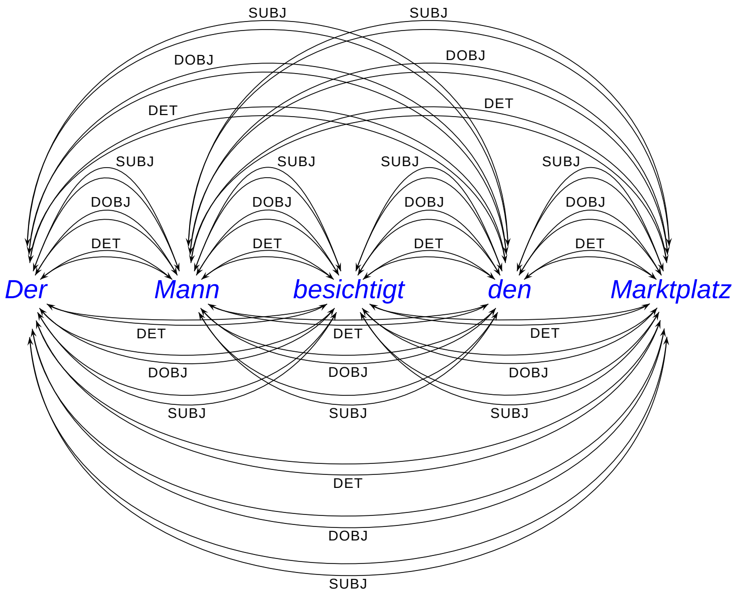
- size of the grammar (English): 2000 Constraints
- quality

	without heuristics	with heuristics
precision	95.5%	97.4%
recall	99.7 ... 99.9%	99.6 ... 99.9%

Dependency parsing as constraint satisfaction

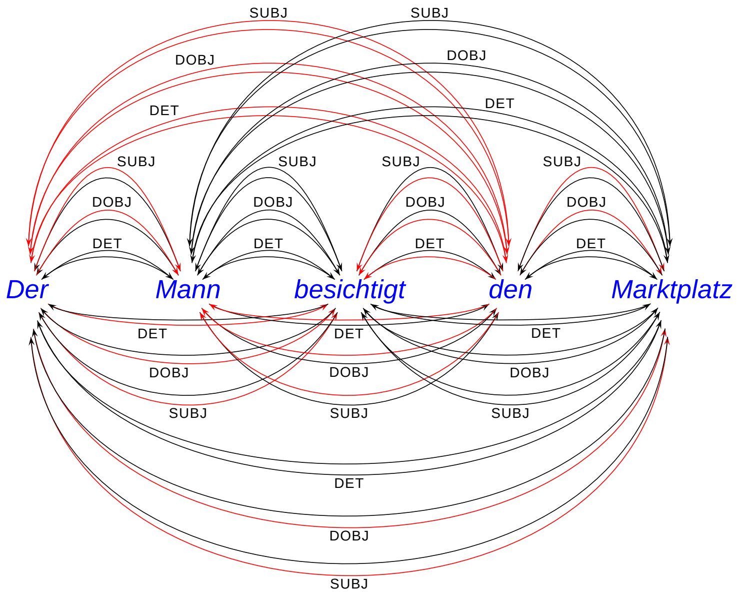
- Constraint Dependency Grammar MARUYAMA 1990
- each word form of a sentence corresponds to a variable.
  $\longrightarrow$ number of variables is a priori unknown.
  $\longrightarrow$ no predefined meaning for variables.
- every constraint must hold for each variable or a combination thereof.
- values are taken from the domain $W \times L$
- constraints license linguistically meaningful structures
- parsing can be understood as structural disambiguation: find a complete variable assignment which satisfies all constraints

Constraining structures



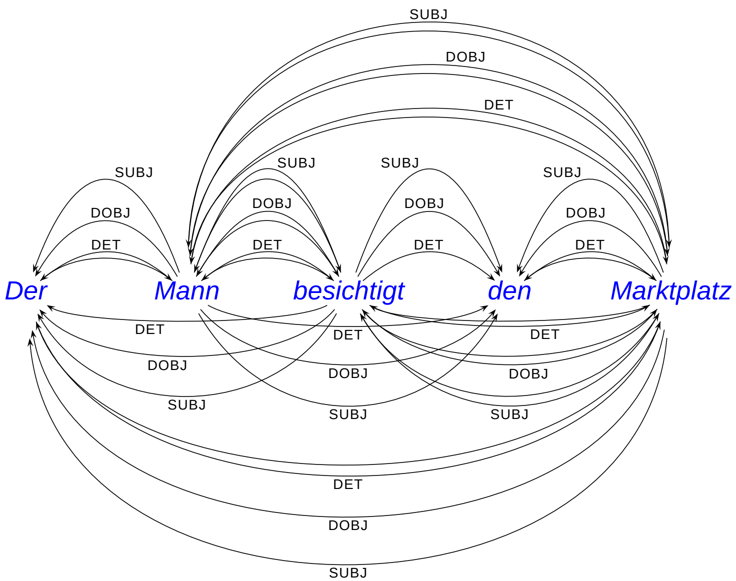
Initial state of a parsing problem with three labels (DET, SUBJ, DOBJ)

Constraining structures

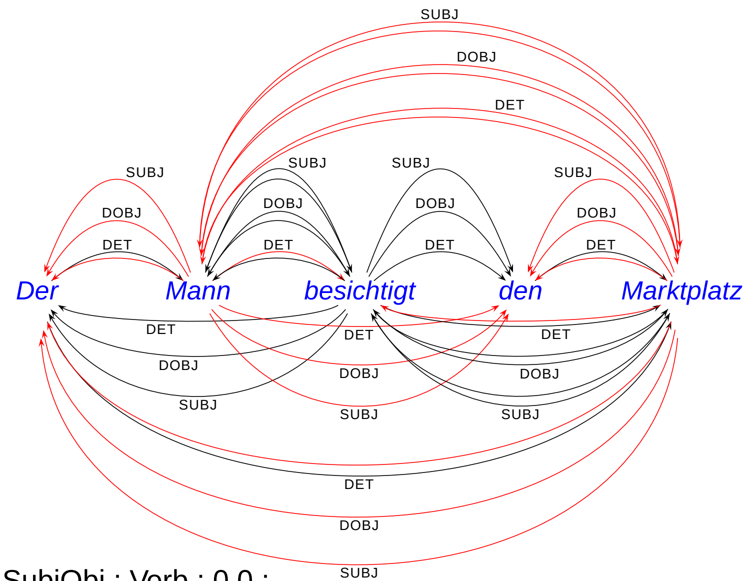


$\{X\}$: DetNom : Det : 0.0 : $X{\downarrow}cat{=}det \;\rightarrow\; X{\uparrow}cat{=}noun \wedge X.label{=}\text{DET}$

Constraining structures

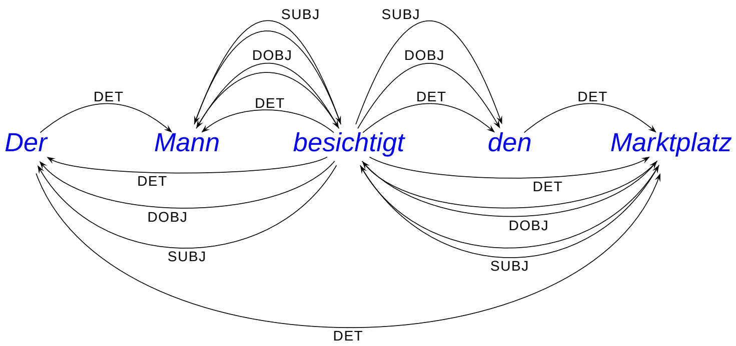


Constraining structures

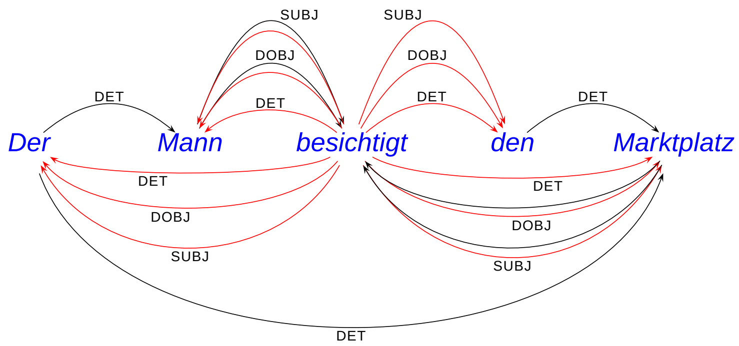


$\{X\}$: SubjObj : Verb : 0.0 :
$X{\downarrow}cat{=}noun \;\rightarrow\; X{\uparrow}cat{=}vfin \wedge X.label{=}\text{SUBJ} \vee X.label{=}\text{DOBJ}$

Constraining structures

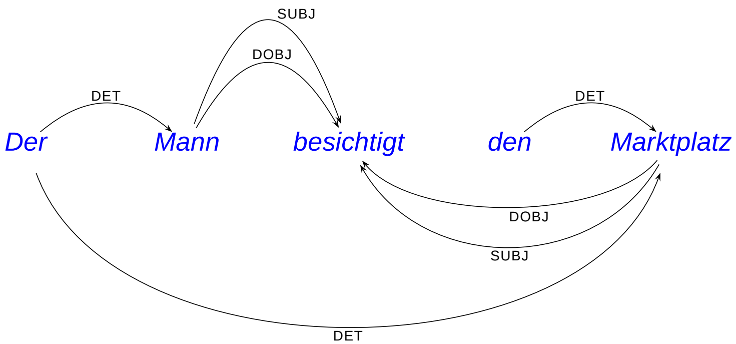


Constraining structures

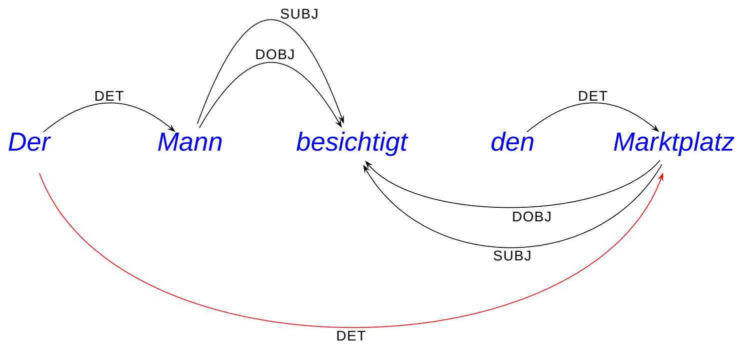


$\{X\}$: Root : Verb : 0.0 :
X$\downarrow$cat=vfin $\rightarrow$ X$\uparrow$cat=nil

Constraining structures

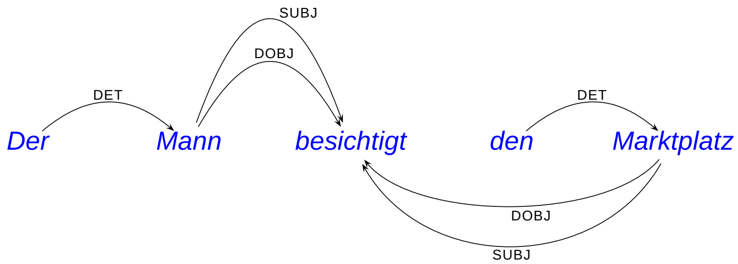


Constraining structures

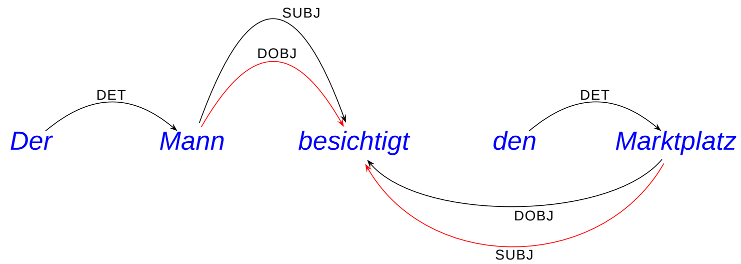


$\{X,Y\}$: Unique : General : 0.0 :
X$\uparrow$id=Y$\uparrow$id $\rightarrow$ X.label$\neq$Y.label

Constraining structures

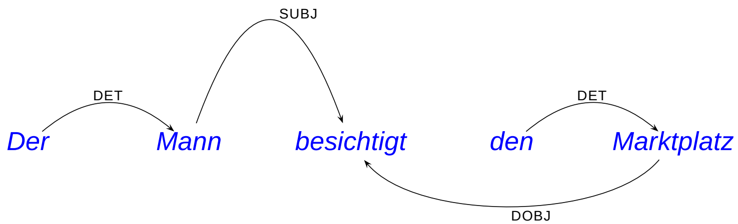


Constraining structures



$\{X,Y\}$: SubjAgr : Subj : 0.0 :
$X.label = \textsc{subj} \wedge Y.label = \textsc{det} \wedge X{\downarrow}id = Y{\uparrow}id \;\rightarrow\; Y{\uparrow}case = Y{\downarrow}case = nom$

Constraining structures



Dependency parsing as constraint satisfaction

- extensions
  - relational view on dependency structures instead of a functional one:
    - → SCHRÖDER (1996): access to lexical information at the modifying *and* the dominating node
  - recognition uncertainty / lexical ambiguity
    - → HARPER AND HELZERMAN (1996): hypothesis lattice additional global constraint (path criterion) introduced
  - access to morphosyntactic features in the lexicon

Dependency parsing as constraint satisfaction

- weighted constraints (penalty factors):
  reduced preference for hypotheses which violate a constraint

$w(c) = 0$	crisp constraints: need always be satisfied e.g. licensing structural descriptions
$0 < w(c) < 1$	weak constraints: may be violated as long as no better alternative is available
$w(c) << 1$	strong, but defeasible well-formedness conditions
$w(c) >> 0$	defaults, preferences, etc.
$w(c) = 1$	senseless, neutralizes the constraint

Dependency parsing as constraint satisfaction

Why weighted constraints?

- Weights help to fully disambiguate a structure.
  - Hard constraints are not sufficient (HARPER ET. AL 1995).
- Many language regularities are preferential and contradictory.
  - extraposition
  - linear ordering in the German mittelfeld
  - topicalization
- Weights are useful to guide the parser towards promising hypotheses.
- Weights can be used to trade speed against quality.

Dependency parsing as constraint satisfaction

- accumulating (multiplying) the weights for all constraints violated by a partial structure
  $\rightarrow$ numerical grading for single dependency relations and pairs of them
- combining local scores by multiplying them into a global one

$$w(t) = \prod_{e \in t} \prod_{c.\text{violates}(e,c)} w(c) \cdot \prod_{(e_i,e_j) \in t} \prod_{c.\text{violates}((e_i,e_j),c)} w(c)$$

- determining the optimal global structure

$$t(s) = \arg\max_t w(t)$$

$\rightarrow$ parsing becomes a constraint optimization problem

Dependency parsing as constraint satisfaction

- writing constraints is counterintuitive
  - CFG: to extend coverage, *add* or *extend* a rule
  - CDG: to extend coverage, *remove* or *weaken* a constraint
- but: the parser itself supports grammar development providing *diagnostic information*
  - constraints violated by the optimal structure are identified

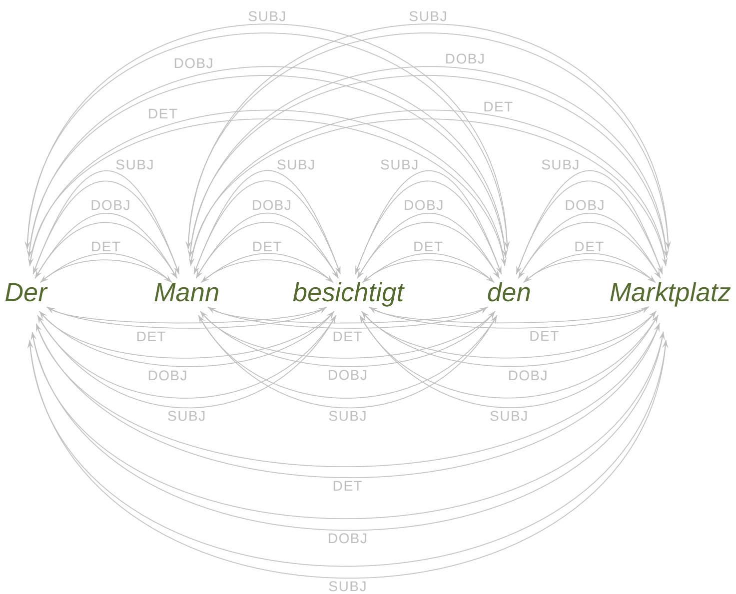
Dependency parsing as constraint satisfaction

- high-arity constraints are expensive
  - → usually at most binary ones are allowed
  - → approximation of constraints with higher arity
- constraint satisfaction is only passive (no value assignment)
  - → approximation of a transitive closure
    e.g. projection, agreement, . . .

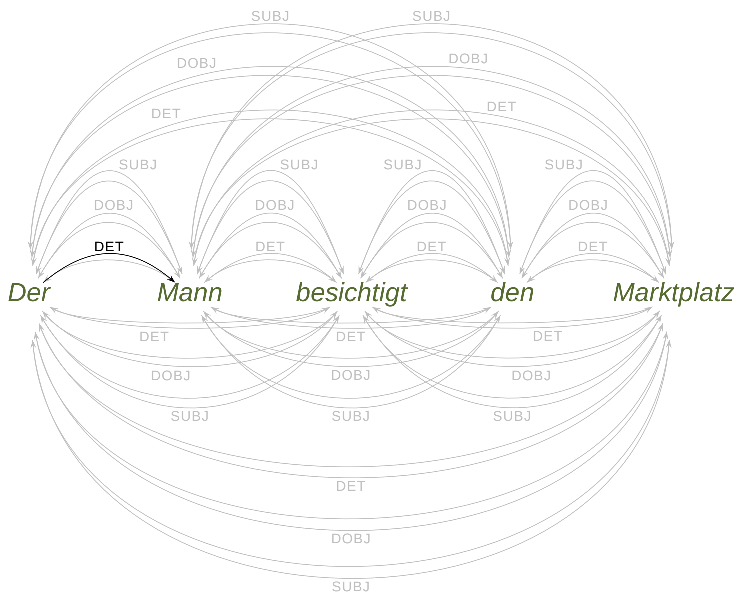
Dependency parsing as constraint satisfaction

- **consistency**: works only for hard constraints
- **pruning**: successively remove the least preferred dependency relations
- **search**: determine the optimum dependency structure
- **structural transformation**: apply local repairs to improve the overall score

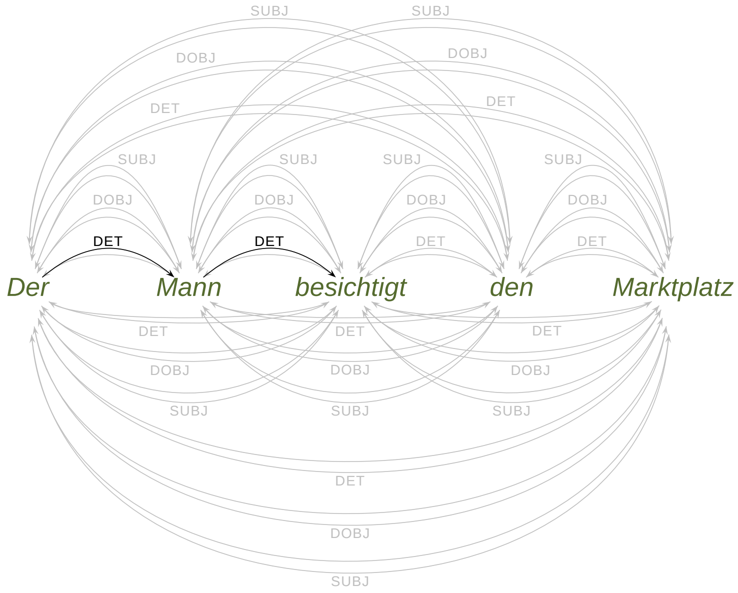
Search



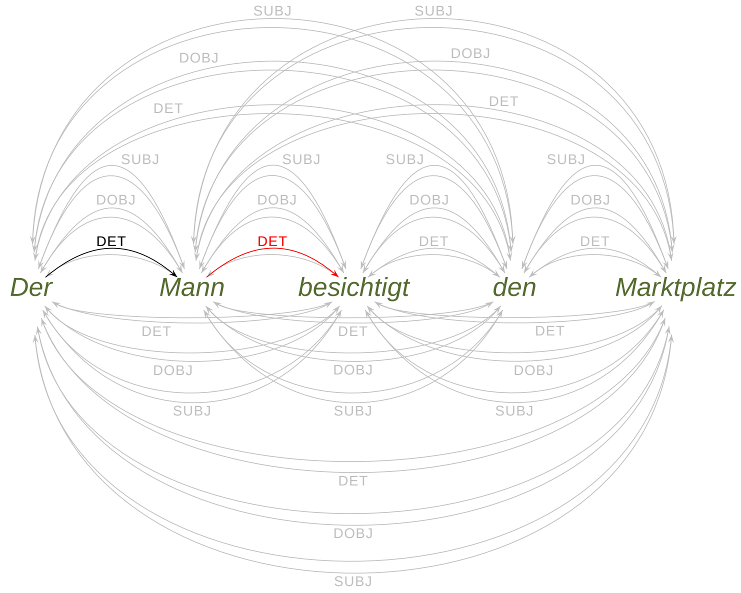
Search



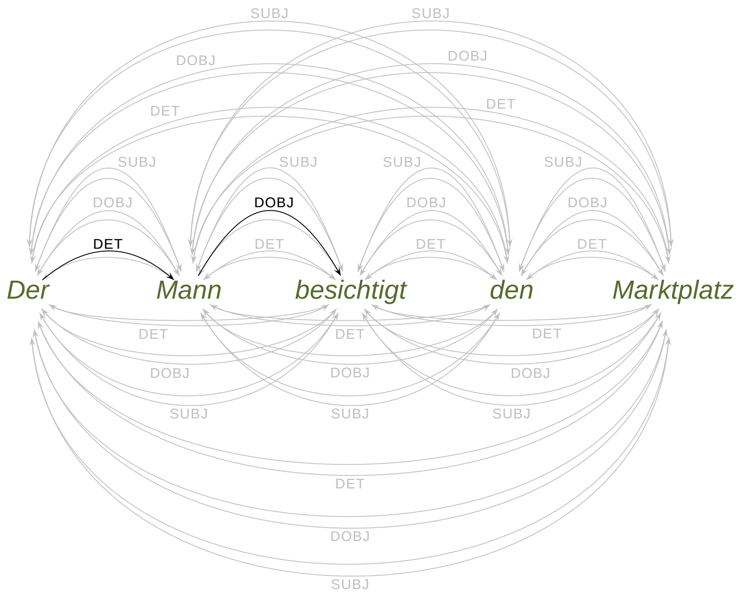
Search



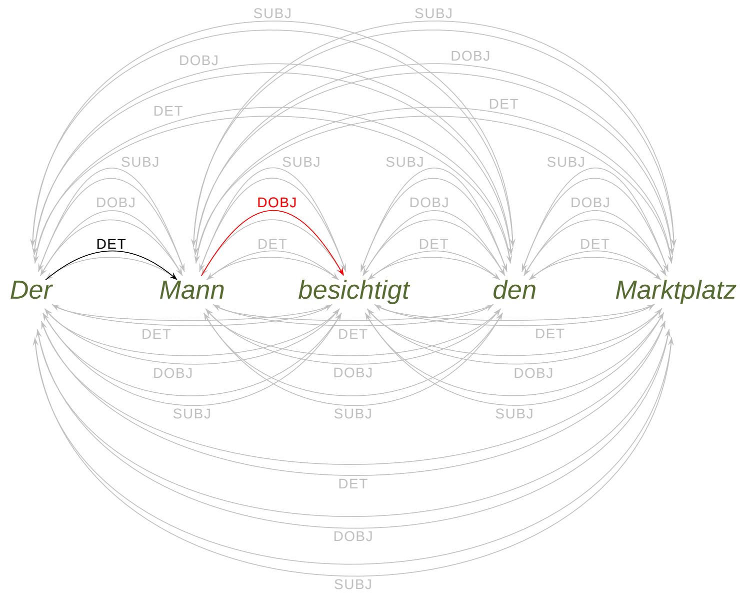
Search



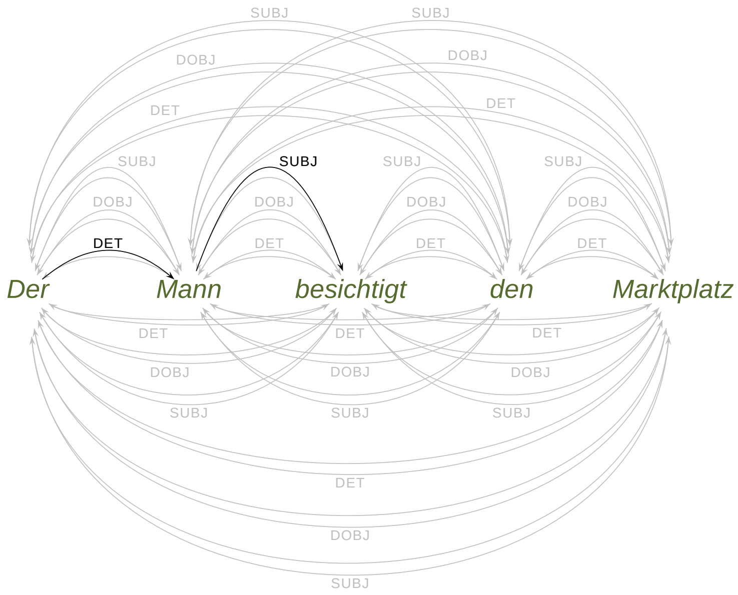
Search



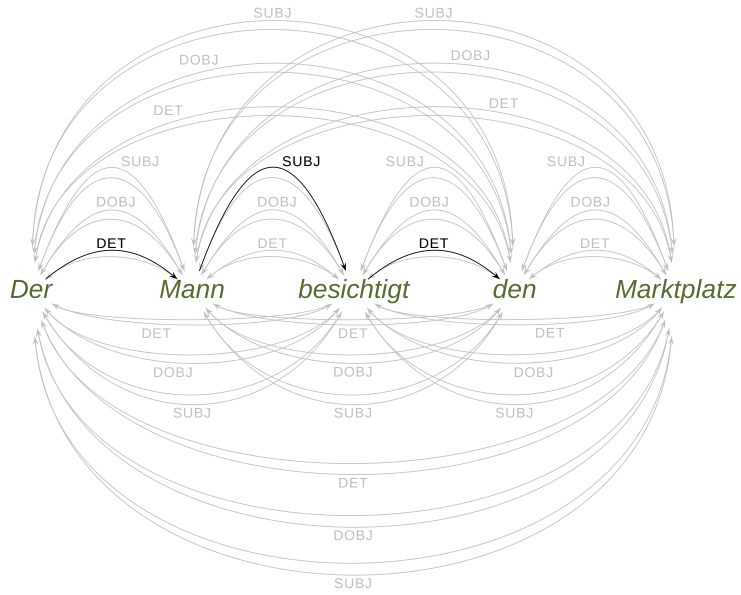
Search



Search



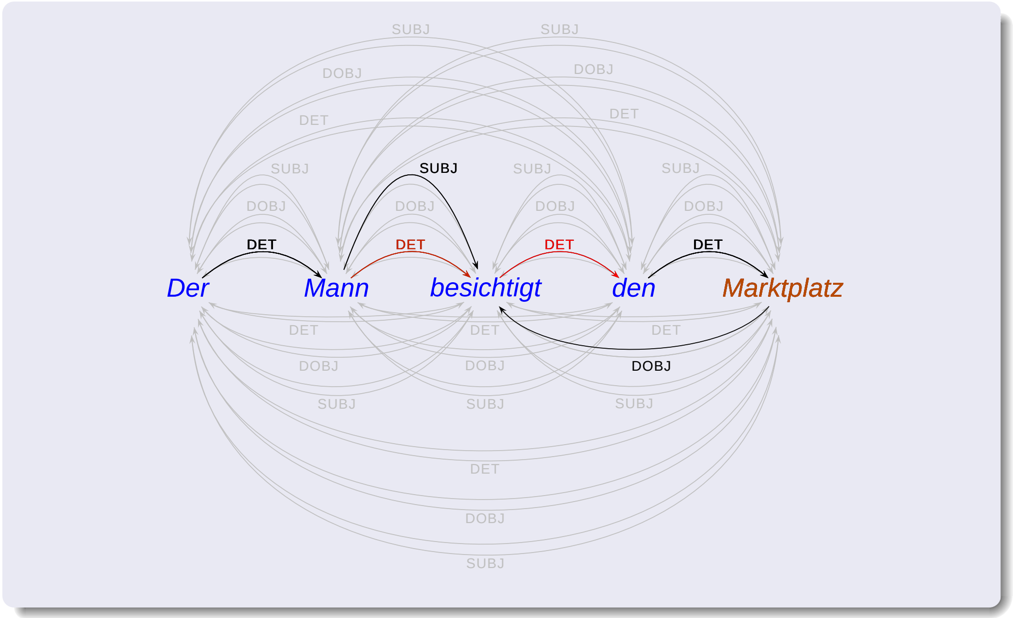
Search



Dependency parsing as constraint satisfaction

- structural transformations: elementary repair operations
  - choose another attachment point
  - choose another edge label
  - choose another lexical reading

Transformation-based parsing



Structural Transformation

- Usually local transformations result in inacceptable structures
  - sequences of repair steps have to be considered.
  - e.g. swapping SUBJ and DOBJ

a)	syntax	. . .
der_1	det/2	. . .
mann_2	**dobj/3**	. . .
besichtigt_3	root/nil	. . .
den_4	den/5	. . .
marktplatz_5	**subj/3**	. . .

$\Longrightarrow$

b)	syntax	. . .
der_1	det/2	. . .
mann_2	**subj/3**	. . .
besichtigt_3	root/nil	. . .
den_4	det/5	. . .
marktplatz_5	**dobj/5**	. . .

Frobbing*

- gradient descent search
- escaping local minima:
  increasingly complex transformations $\rightarrow$ local search
- heuristically guided tabu search
  - transformation with perfect memory
  - propagation of limits for the score of partial solutions
- faster than best-first search for large problems
- inherently anytime

**frobbing*: randomly adjusting the settings of an object, such as the dials on a piece of equipment or the options in a software program. (The Word Spy)

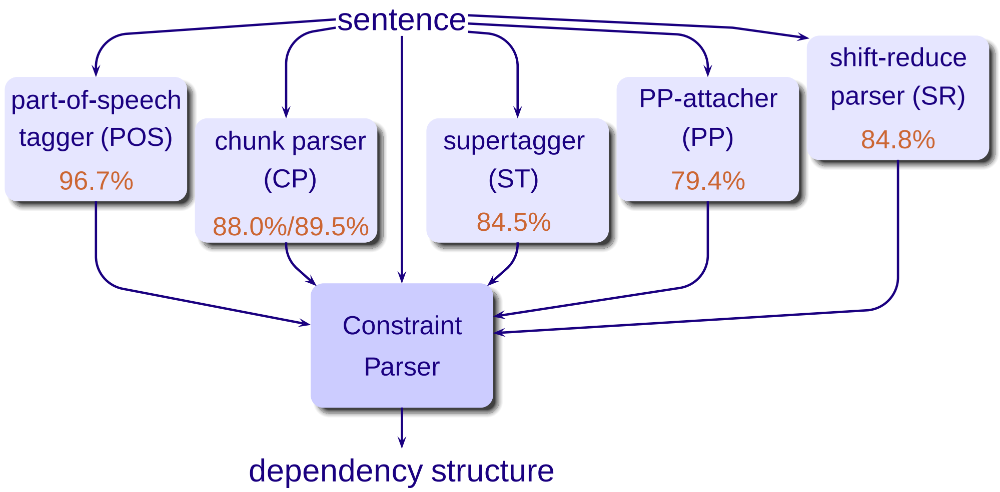
Solution Methods

	sound-ness	complete-ness	efficiency	predicta-bility	interrupt-ability	termi-nation
pruning	$--$	$--$	$+/-$	$++$	$--$	$++$
search	$++$	$+$	$--$	$--$	$--$	$++$
transformation	$+$	$-$	$-$	$+$	$++$	$-$

Hybrid parsing

- the bare constraint-based parser itself is weak
- but: constraints can be used as interface to external predictor components
- predictors are all probabilistic, thus inherently unreliable
  $\rightarrow$ can their information still be useful?
- several predictors $\rightarrow$ consistency cannot be expected

Hybrid parsing



Hybrid parsing

- results on a 1000 sentence newspaper testset (FOTH 2006)

Predictors	accuracy unlabelled	accuracy labelled
0: none	72.6%	68.3%
1: POS only	89.7%	87.9%
2: POS+CP	90.2%	88.4%
3: POS+PP	90.9%	89.1%
4: POS+ST	92.1%	90.7%
5: POS+SR	91.4%	90.0%
6: POS+PP+SR	91.6%	90.2%
7: POS+ST+SR	92.3%	90.9%
8: POS+ST+PP	92.1%	90.7%
9: all five	92.5%	91.1%

- net gain although the individual components are unreliable

Hybrid parsing

- robust across different corpora (FOTH 2006)

text type	sentences	average length	accuracy unlabelled	accuracy labelled
law text	1145	18.4	90.7%	89.6%
online news	10000	17.3	92.0%	90.9%
Bible text	2709	15.9	93.0%	91.2%
trivial literature	9547	13.8	94.2%	92.3%

skip

Relative Importance of Information Sources

Class	Purpose	Example	Importance
agree	rection and agreement	subjects have nominative case	1.02
cat	category cooccurrence	prepositions do not modify each other	1.13
dist	locality principles	prefer the shorter of two attachments	1.01
exist	valency	finite verbs must have subjects	1.04
init	hard constraints	appositions are nominals	3.70
lexical	word-specific rules	“entweder” requires following “oder”	1.02
order	word-order	determiners precede their regents	1.11
pos	POS tagger integration	prefer the predicted category	1.77
pref	default assumptions	assume nominative case by default	1.00
proj	projectivity	disprefer nonprojective coordinations	1.09
punc	punctuation	subclauses are marked with commas	1.03
root	root subordinations	only verbs should be tree roots	1.72
sort	sortal restrictions	“sein” takes only local predicatives	1.00
uniq	label cooccurrence	there can be only one determiner	1.00
zone	crossing of marker words	conjunctions must be leftmost dependents	1.00

Relative Importance of Information Sources

Class	Purpose	Example	Importance
init	hard constraints	appositions are nominals	3.70
pos	POS tagger integration	prefer the predicted category	1.77
root	root subordinations	only verbs should be tree roots	1.72
cat	category cooccurrence	prepositions do not modify each other	1.13
order	word-order	determiners precede their regents	1.11
proj	projectivity	disprefer nonprojective coordinations	1.09
exist	valency	finite verbs must have subjects	1.04
punc	punctuation	subclauses are marked with commas	1.03
agree	rection and agreement	subjects have nominative case	1.02
lexical	word-specific rules	“entweder” requires following “oder”	1.02
dist	locality principles	prefer the shorter of two attachments	1.01
pref	default assumptions	assume nominative case by default	1.00
sort	sortal restrictions	“sein” takes only local predicatives	1.00
uniq	label cooccurrence	there can be only one determiner	1.00
zone	crossing of marker words	conjunctions must be leftmost dependents	1.00

Selling Points

- robustness against ungrammatical input
- inherent diagnostic abilities:
  constraint violations can be interpreted as error diagnoses
  - transformation-based parsing is conflict-driven
  - crucial for interactive grammar development
  - applications for second language learning
- inherent anytime properties
  - interruptable
  - processing time can be traded for parsing accuracy

Selling Points

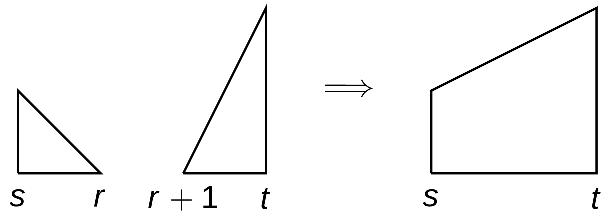
- framework for soft information fusion
  - syntax, semantics, information structure, ...
  - shallow processing components
- achieves always full disambiguation
- partial results can be obtained if needed
- you have to be **very** patient

Structure-based dependency parsing

- MST-parser (McDonald)
- large margin learning → scoring candidate edges
- first order (unary) / second order (binary) constraints
- two step approach:
  - computation of bare attachments
  - labellings as edge classification
- problem: combining second order constraints and non-projective parsing
- projective tree building: Eisner (1996)
  - parse the left and the right dependents independently
  - join the partial trees later

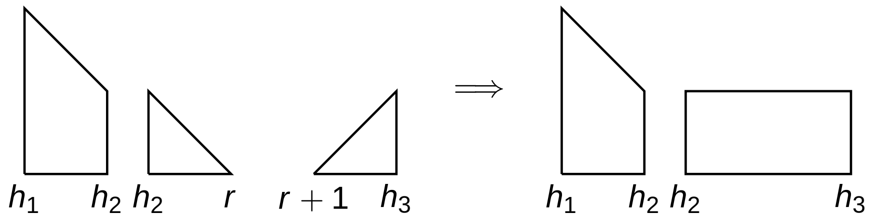
Structure-based dependency parsing

- to build an incomplete subtree from word index s to t find a word index r ($s \leq r < t$) which maximizes the sum of the scores of the two complete subtrees plus the score of the edge from s to t



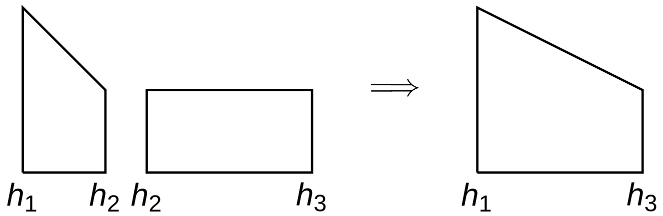
Structure-based dependency parsing

- extension to second order constraints:
  - establishing a dependency in two phases
  - sibling creation + head attachment
- to establish an edge between h_3 and h_1, given that an edge between h_2 and h_1 had already been established, find a word index r ($h_2 \leq r < h_3$) that maximizes the score of making h_2 and h_3 sibling nodes



Structure-based dependency parsing

- delay the completion of an item until all the sibling nodes have been collected



Structure-based dependency parsing

- re-evaluation of MST on the WCDG annotations
- with interpunction

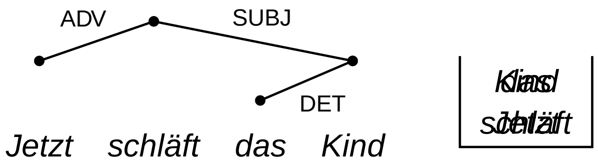
	accuracy[%]	
	structural	labelled
MST parser	91.9	89.1
WCDG (POS tagger only)	89.7	87.9
WCDG (all predictors)	92.5	91.1

- without interpunction

	accuracy[%]	
	structural	labelled
MST on NEGRA	90.5	87.5
MST on TIGER (CoNLL 2006)	90.4	87.3

History-based dependency parsing

- MaltParser NIVRE (2004): choice between four parser actions:
  shift / left-attach + reduce / right-attach + shift / reduce



- support vector machine trained on the parse history to predict the best next parser action
- parser takes deterministic decisions: eager processing
- fully left-to-right incremental processing

Parser combination

- WCDG + MST-Parser
- Reparsing (MST-Parser + Malt-Parser)
- Retraining (MST-Parser + Malt-Parser)

Parser combination

- WCDG has proven useful to integrate external predictor
- so far, all predictors consider
  - partial aspects of the parsing problem
    tagger, supertagger, pp-attacher, ...,
  - or use a different representation
    projective vs. non-projective
- What happens ...
  - ... if two parsers for exactly the same task are combined?
  - ... if the predictor becomes superior?

Parser Combination

- using the output of MST to guide WCDG
- three additional constraints
  - Is the modifiee the same?
  - Is the root node the same?
  - Is the label the same?
- separate constraint weights for attachment and label

Parser Combination

- results

	accuracy[%] with interpunction		accuracy[%] without interpunction	
	structural	labelled	structural	labelled
MST parser	91.9	89.1	89.5	86.0
WCDG (POS tagger only)	89.7	87.9	88.0	86.0
WCDG (all predictors)	92.5	91.1	91.3	90.0
WCDG + POS tagger + MST	93.1	91.8		
WCDG + all predictors	93.9	92.6	92.9	91.4

- high degree of synergy

Phrase structure parsing

- phrase structures
- parsing strategies
- chart parsing
- probabilistic models
- restricted phrase structure models

Phrase structure

- constituents as basic units
- constituents are embedded into other constituents
- constituent structure can be described by means of a context free grammar
  - non-terminal symbols: S, NP, VP, PP, ...
  - terminal symbols: *waits, for, in, the, John, Mary, park*

  NT-Symbol $\longrightarrow$ {T-Symbol $|$ NT-Symbol}*
- rule application
  - generatively
  - analytically
- parser has to accomplish three tasks
  - computing the attachment, the label, and the extension of a phrase

Phrase structure

- phrase structure tree is a byproduct of the derivation process (recursive rule application)

  $\longrightarrow$ close relationship between
  - rule structure
  - structural description
  - rule application (analysis/generation)
- rules can be extracted from a given phrase structure tree

Phrase structure

- lexical insertion rules, preterminal rules, lexicon

  N $\rightarrow$ *Mary*
  N $\rightarrow$ *John*
  N $\rightarrow$ *park*
  P $\rightarrow$ *in*
  D $\rightarrow$ *the*
  V $\rightarrow$ *sees*

Phrase structure

- structure-building rules, grammar

  S $\rightarrow$ NP VP
  VP $\rightarrow$ V NP
  VP $\rightarrow$ V PP
  VP $\rightarrow$ V PP PP
  PP $\rightarrow$ P NP
  NP $\rightarrow$ N

- first constraint on possible forms of rules
  - lexicon
    PT-Symbol $\rightarrow$ T-Symbol
  - grammar
    NT-Symbol $\rightarrow$ {NT-Symbol | PT-Symbol}*

Phrase structure

- recursive rules: potentially infinitely many sentences can be generated
  $\rightarrow$ creativity of language competence
- goal of linguistic modelling: specification of additional constraints on the possible rule forms

Phrase structure

- phrasal categories: distributional type (purely structural perspective)

- phrasal categories are derived from lexical ones by adding additional constituents

  N $\Rightarrow$ NP
  V $\Rightarrow$ VP
  A $\Rightarrow$ AP
  ADV $\Rightarrow$ ADVP
  P $\Rightarrow$ PP

Parsing strategies

- rule application from left to right: top-down analysis
  - derivation of a sentence from the start symbol

    S
    NP VP
    N V NP
    John sees NP
    John sees Mary
- rule application from right to left: bottom up analysis
  - derivation of the start symbol from the sentence:

    John sees Mary
    N V N
    NP V NP
    NP VP
    S

Parsing strategies

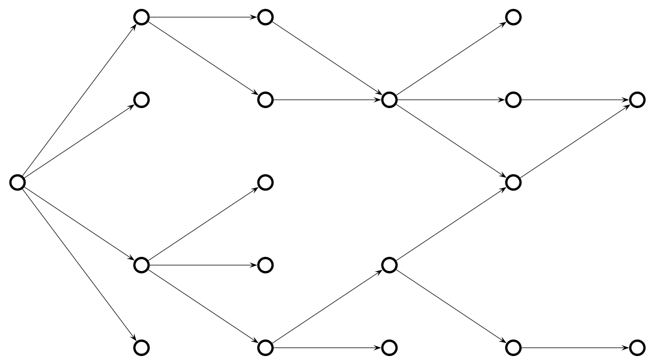
- all alternatives for rule applications need to be checked
- ambiguities do not allow local decisions
- lexical ambiguities: *green/VINF/VFIN/NN/ADJ/ADV*
- structural ambiguities as a consequence of lexical ones

Parsing strategies

- purely structural ambiguities
  [$_{NP}$ the man [$_{PP}$ with the hat [$_{PP}$ on the stick]]]
  [$_{NP}$ the man [$_{PP}$ with the hat] [$_{PP}$ on the stick]]
  . . . , weil [$_{NP}$ dem Sohn des Meisters] [$_{NP}$ Geld] fehlt.
  . . . , weil [$_{NP}$ dem Sohn] [$_{NP}$ des Meisters Geld] fehlt.
- local ambiguities can be resolved during subsequent analysis steps
- global ambiguities remain until the analysis finishes

Parsing strategies

- parsing as search
  - alternative rule applications create a search space



Parsing strategies

- expectation driven (top-down, expand-reduce)
  - problem: left/right recursive rules cause termination problems
    - even in case of indirect recursion:

      X → Y a
      Y → X
  - solution: transformation into a weakly equivalent grammar without left/right recursion
    - linguistically motivated derivation structure is lost
    - workaround: generating a separated structure by means of unification

Parsing strategies

- data driven (bottom-up, shift-reduce)
  - problem: empty productions (linguistically motivated)

    X → ϵ
    - perhaps ”licensing” empty categories by lexical nodes
  - problem: unary rules which form a cycle
    - avoid them completely

Parsing strategies

- depth-first
  - alternative rule applications are tried later on
  - storing them on a stack
- breadth-first
  - alternative rule applications are tried in ”parallel”
  - maintaining the alternatives in a queue

Parsing strategies

- left-to-right
  - input is processed beginning from its left side
- right-to-left
  - input is processed beginning from its right side

Parsing strategies

- mixed strategies
  - Left-Corner-Parsing: top-down analysis activating a rule by its left corner
  - robust parsing for erroneous input:
    bottom-up analysis and subsequent top-down reconstruction in case of failure (MELLISH 1989)
  - island parsing: bidirectional analysis starting from reliable hypotheses (e.g. for speech recognition results)

Chart parsing

- effciency problem: repetition of analysis steps on alternative analysis paths
- recombination of search paths is required
- data
  - German with head-final verb group
  - unmarked case: subclause ordering

  ..., *weil der Vater seine Kinder liebt*.
  ..., *weil der Vater seinen Kindern glaubt.*
  ..., *weil der Vater seinen Kindern ein Eis versprach.*
  ..., *weil der Vater seinen Kindern mit einer Strafe droht.*

Chart parsing

- grammar

  S' $\longrightarrow$ Konj S
  S $\longrightarrow$ NP_n VP
  VP $\longrightarrow$ NP_a V_a
  VP $\longrightarrow$ NP_d V_d
  VP $\longrightarrow$ NP_d NP_a $V_{d,a}$
  VP $\longrightarrow$ NP_d $PP_{mit,d}$ $V_{d,mit}$
  NP_X $\longrightarrow$ D_X N_X
  $PP_{X,Y}$ $\longrightarrow$ P_X NP_Y
- Example analysis: top-down, depth-first
  ... der Vater seinen Kindern ein Eis versprach.

Chart parsing

S
NP_n VP
D_n N_n VP
d N_n VP
d v VP

d v NP_d V_d
d v D_d N_d V_d
d v s N_d V_d
d v s k V_d

d v NP_a V_a
d v D_a N_a V_a

d v NP_d NP_a $V_{d,a}$
d v D_d N_d NP_a $V_{d,a}$
d v s N_d NP_a $V_{d,a}$
d v s k NP_a $V_{d,a}$
d v s k D_a N_a $V_{d,a}$
d v s k e N_a $V_{d,a}$
d v s k e e $V_{d,a}$
d v s k e e v

...

Chart parsing

- well-formed substring table (chart)
  - directed acyclic graph (DAG) with
    - one source (beginning of the sentence)
    - one sink (end of the sentence) and
    - a total precedence relation on the nodes
  - edges correspond to successfully recognized constituents

Chart parsing

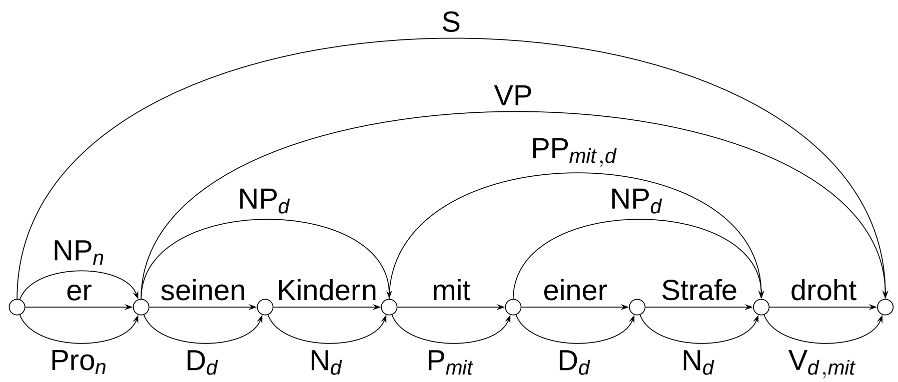


Chart parsing

	1	2	3	4	5	6	7
0	er Pro_n NP_n						S
1		seinen D_d	NP_d				VP
2			Kindern N_d				
3				mit P_{mit}		PP_{mit}	
4					einer D_d	NP_d	
5						Strafe N_d	
6							droht $V_{d,mit}$

Chart parsing

- Cocke-Younger-Kasami algorithm (KASAMI 1965, YOUNGER 1967)
- grammar in Chomsky-normalform
  - binary branching rules: X $\longrightarrow$ Y Z
  - pre-terminal/lexical rules: X $\longrightarrow$ a

Chart parsing

- properties of the CYK algorithm
  1. length o the derivation is constant:
     n lexical rules + n-1 binary branching rules
  2. number of binary partitionings of a sentence is constant: n-1

     ```
     ((a) (b c d))
     ((a b) (c d))
     ((a b c) (d))
     ```
  3. no structural ambiguities due to different segmentations of the sentence

     VP $\rightarrow$ NP NP V
     VP $\rightarrow$ NP V
     VP $\rightarrow$ V

Tabellenparsing

- CYK algorithm
  1. initialisaton of the table

     for $i = 0$ to $n - 1$:
       CHART$_{i,i+1} \Leftarrow \{$ X | X $\in$ V$_T$ and w$_{i+1} \in$ X $\}$
  2. computation of the remaining entries

     for $k = 2$ to n:
       for $i = 0$ to $n - k$:
         $j \Leftarrow i + k$
         CHART$_{i,j} \Leftarrow \{$ A | (A $\rightarrow$ X Y) $\in$ R $\wedge \exists$ m . (X $\in$ CHART$_{i,m}$
                 $\wedge$ Y $\in$ CHART$_{m,j}$,   mit $i < m < j$ $\}$
     if S $\in$ CHART$_{0,n}$
       then RETURN(*true*)
       else RETURN(*false*)

Chart parsing

- bottom-up analysis
  - time complexity $\mathcal{O}(n^3)$
  - memory complexity $\mathcal{O}(n^2)$
  - achieved by reycling of intermediate results (recombination)
- disadvantage: still constituents are generated which cannot be integrated into a larger structure (dead ends)
  $\rightarrow$ EARLEY parser

Chart parsing

- active chart
  - extension: even incomplete attempts of rule applications are recorded in the chart
  - active edges:
    open expectations for the right context
    notation: $\langle$ a, b, A $\rightarrow$ B . C D $\rangle$
  - inactive edges:
    completely satisfied expectations for the right context
    notation: $\langle$ a, b, A $\rightarrow$ B C D . $\rangle$

Chart parsing

- TD rule (initialisation)

  For all rules A $\rightarrow$ w_1 where A is a start symbol of the grammar, add an edge $\langle$ 0, 0, A $\rightarrow$. w_1 $\rangle$ to the chart.
- rule: S $\rightarrow$ NP_n VP

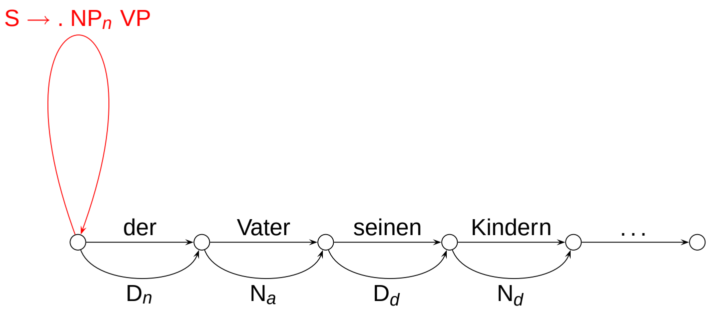


Chart parsing

- TD-rule (edge introduction)

  When adding a rule $\langle$ i, j, A $\rightarrow$ w_1 . B w_2 $\rangle$ to the chart, add for each rule B $\rightarrow$ w_3 an edge $\langle$ j, j, B $\rightarrow$. w_3 $\rangle$.
- rule: NP_X $\rightarrow$ D_X N_X

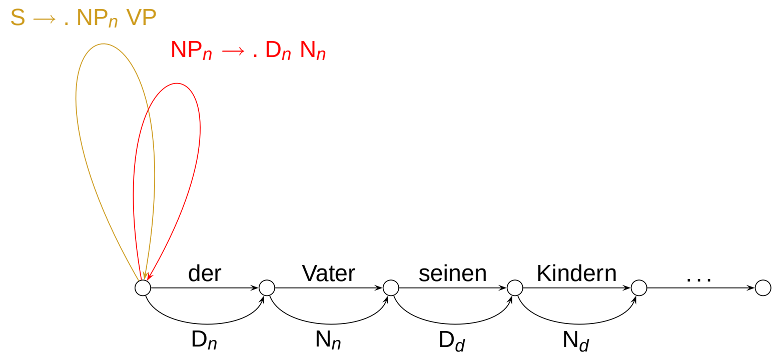


Chart parsing

- fundamental rule (edge expansion)

  If the chart contains two edges $\langle$ i, j, A $\rightarrow$ w_1 . B w_2 $\rangle$ and $\langle$ j, k, B $\rightarrow$ w_3 . $\rangle$, add a third edge $\langle$ i, k, A $\rightarrow$ w_1 B . w_2 $\rangle$.

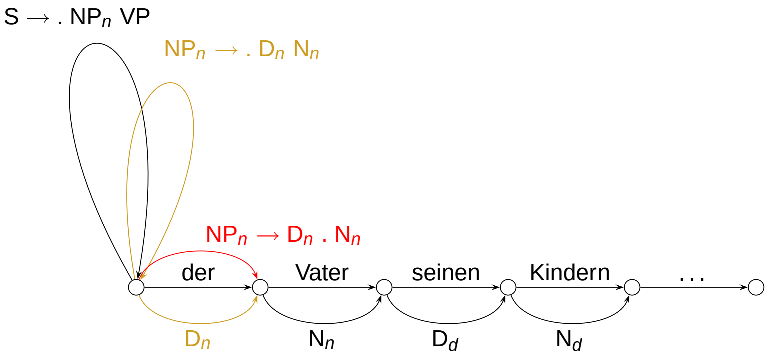


Chart parsing

- repeated application of the fundamental rule

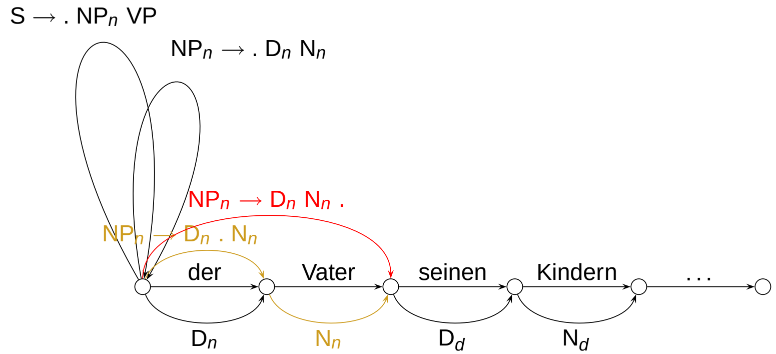


Chart parsing

- repeated application of the fundamental rule

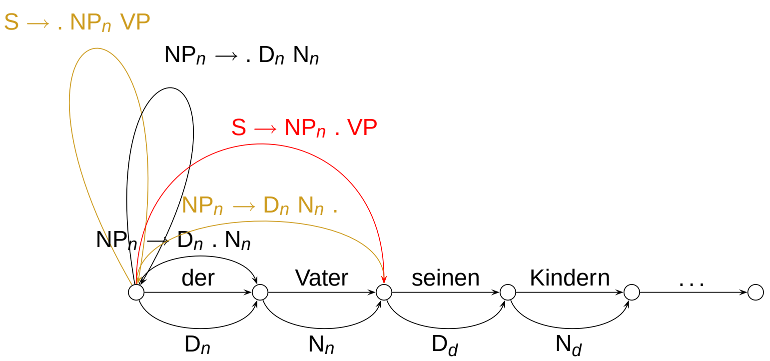


Chart parsing

- repeated application of the top-down rule

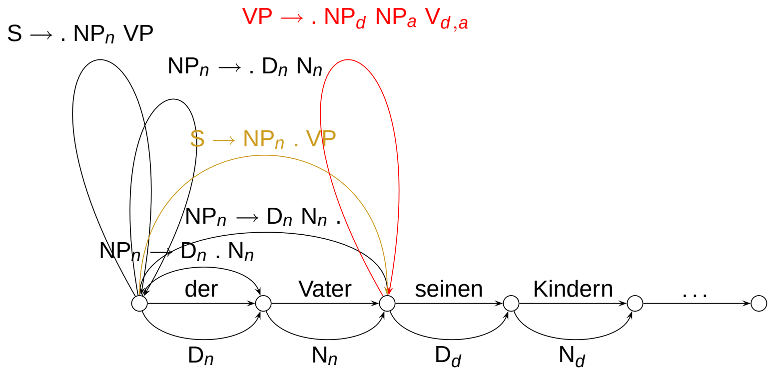


Chart-Parsing

- repeated application of the top-down rule

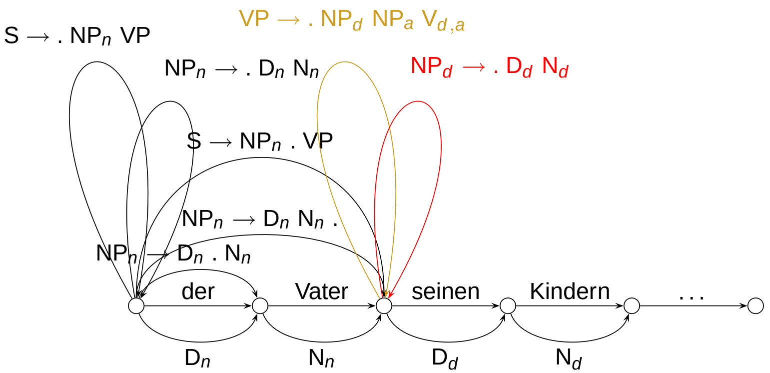


Chart parsing

- repeated application of the fundamental rule

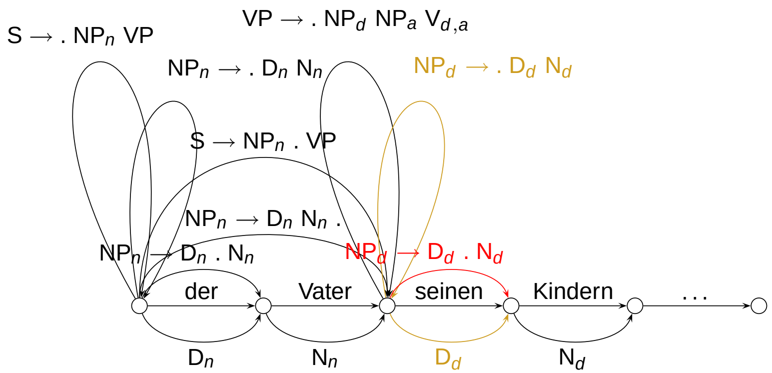


Chart parsing

- repeated application of the fundamental rule

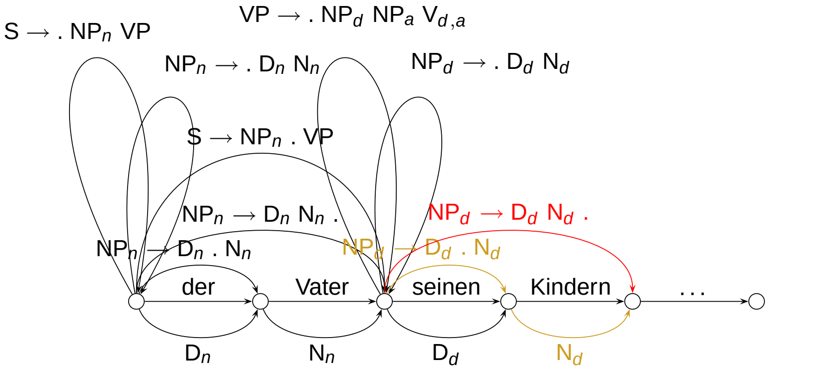


Chart parsing

- repeated application of the fundamental rule

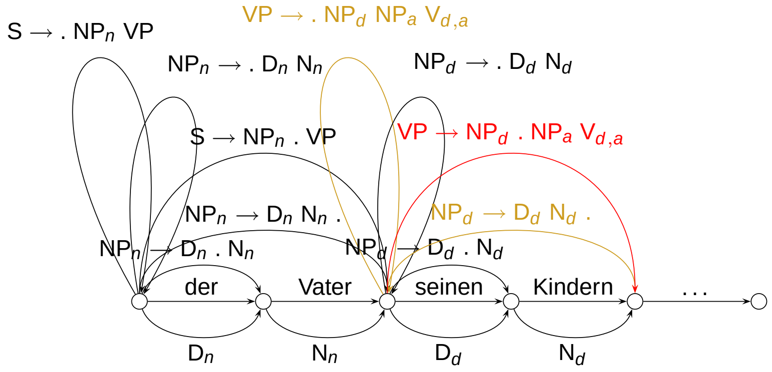


Chart parsing

- repeated application of the top-down rule

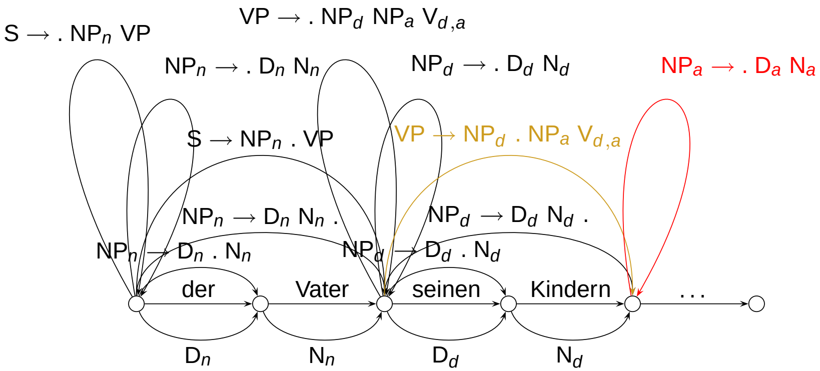


Chart parsing

- Earley algorithm (Earley 1970)
  - for arbitrary context free grammars
    - including recursion, cycles and ϵ-rules
  - mixed top-down/bottom-up strategy, to avoid adding of edges (constituents) which cannot be incorporated into larger ones
    1. top-down condition:
       only edges are added for which the left context is compatible with the requirements of the grammar
    2. bottom-up condition:
       the already applied part of the rule is compatible with the input data

Chart parsing

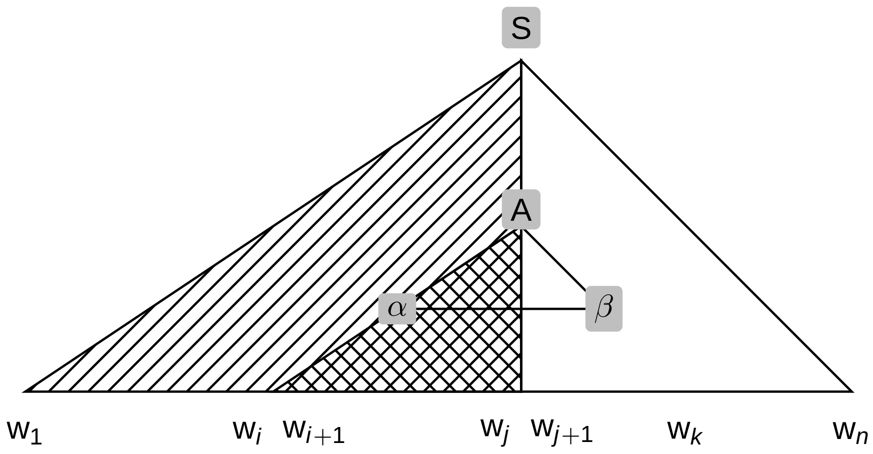


Chart parsing

- elementary operations
  - expand (top-down rule, edge introduction)
  - complete (fundamental rule, edge expansion)
  - shift (introduction of lexical edges)
- different search strategies (depth-first/breadth-first/best-first) are possible depending on the agenda management

Chart parsing

- EARLEY-Algorithmus
  1. initialization

     for all (S $\rightarrow \beta$) $\in$ R: $\text{CHART}_{0,0} \Leftarrow \langle \text{S}, \emptyset, \beta \rangle$

     Apply EXPAND to the previously generated edges until no new edges can be added.
  2. computation of the remaining edges

     for $j = 1, \ldots, n$:
       for $i = 0, \ldots, j$:
         compute $\text{CHART}_{i,j}$:
           1. apply SHIFT to all relevant edges in $\text{CHART}_{i,j-1}$
           2. apply EXPAND and COMPLETE until no new edges can be produced.
     if $\langle \text{S}, \beta, \emptyset \rangle \in \text{CHART}_{0,n}$
       then RETURN(*true*) else RETURN(*false*)

Chart parsing

- a chart-based algorithm is only a recognizer
- extending it to a real parser:
  - extraction of structural descriptions (trees, derivations) from the chart in a separate step
  - basis: maintaining a pointer from an edge to the activating edge in the fundamental rule
  - "collecting" the trees starting with all inactive S-edges

Chart parsing

- time complexity
  - $\mathcal{O}(n^3 \cdot |G^2|)$
  - for deterministic grammars: $\mathcal{O}(n^2)$
  - in many relevant cases: $\mathcal{O}(n)$
- complexity result is only valid for constructing the chart
- tree extraction might require exponential effort in case of exponentially many results

Chart parsing

- space complexity
  - $\mathcal{O}(n^2)$
  - due to the reuse of intermediate results
    - holds only for atomic non-terminal symbols
- chart is a general data structur to maintain intermediate results during parsing
  - alternative parsing strategies are possible
  - e.g. bottom-up

Chart parsing

- bottom-up rule (edge introduction)

  When adding a rule $\langle$ i, j, B $\longrightarrow$ w_1 $\rangle$ for every rule A $\longrightarrow$ B w_2 add another edge $\langle$ i, i, A $\longrightarrow$. B w_2 $\rangle$

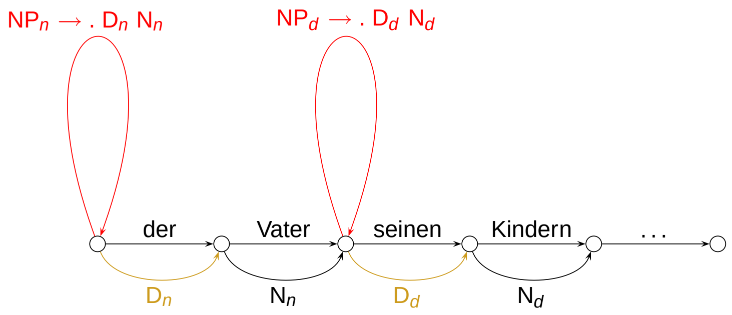


Chart parsing

- application of the fundamental rule

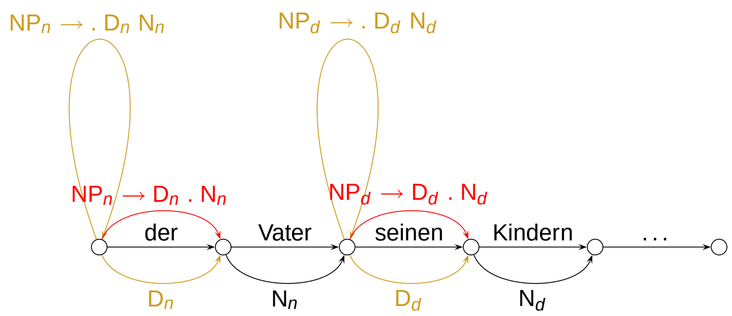


Chart parsing

- application of the fundamental rule

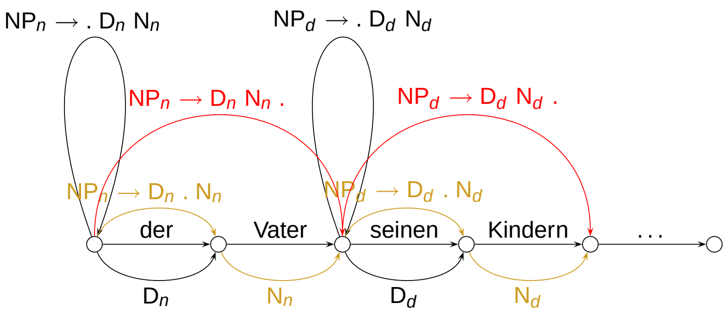


Chart parsing

- Application of the bottom-up rule

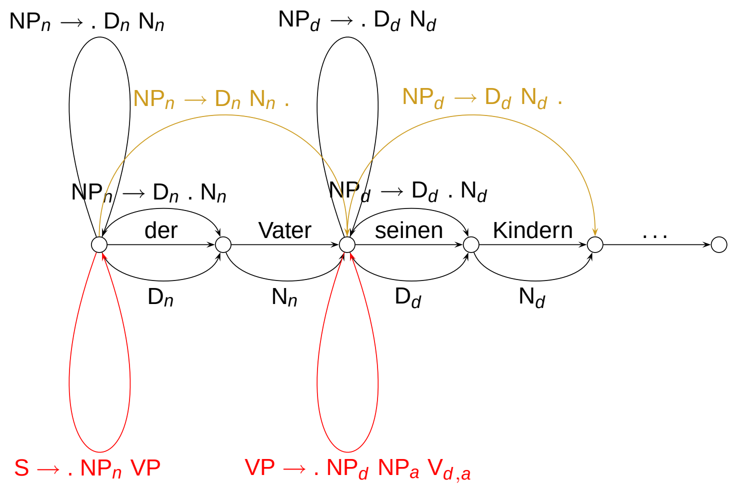


Chart parsing

- application of the fundamental rule

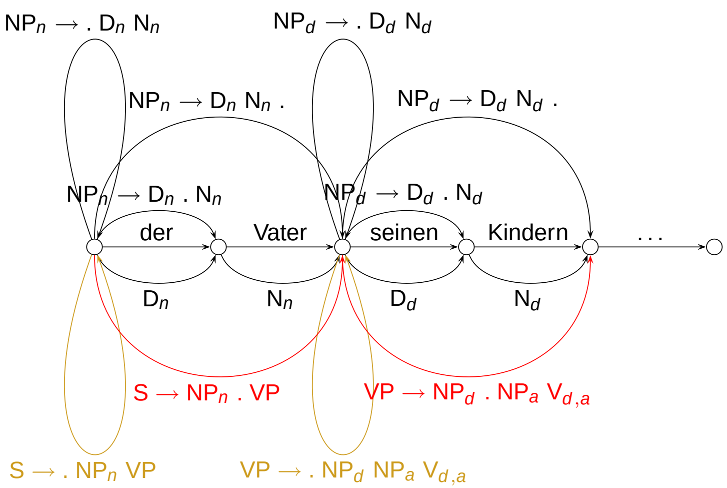


Chart parsing

- parsing is a monotonic procedure of information gathering
  - edges are never deleted from the chart
  - even unsuccessful rule applications are kept
  - edges which cannot be expanded further
- duplicating analysis effort is avoided
  - edge is only added to the chart if not already there

Chart parsing

- agenda
  - list of active edges
  - can be sorted according to different criteria
  - stack: depth-first
  - queue: breadth-first
  - TD-rule: expectation-driven analysis
  - BU-rule: data -driven analysis

Chart parsing

- flexible control for hybrid strategies
- left-corner parsing
  - TD-parsing, but only those rules are activated, which can derive a given lexical category (left corner) directly or indirectly
  - mapping between rules and their possible left corners is computed from the grammar at compile time
  - variant: head-corner parsing

Chart parsing

- best-first parsing
  - sorting the agenda according to confidence values
    - hypothesis scores of speech recognition
    - rule weights (e.g. relative frequency in a tree bank)

Stochastic models

- common problem of all purely symbolic parser
  - high degree of output ambiguity
  - even in case of (very) fine-grained syntactic modelling
  - despite of a dissatisfyingly low coverage
- coverage and degree of output ambiguity are typically highly correlated

Stochastic models

- output ambiguity
  - *Hinter dem Betrug werden die gleichen Täter vermutet, die während der vergangenen Tage in Griechenland gefälschte Banknoten in Umlauf brachten.*
  - *The same criminals are supposed to be behind the deceit who in Greece over the last couple of days brought falsified money bills into circulation.*
  - Paragram (KUHN UND ROHRER 1997): 92 readings
  - Gepard (LANGER 2001): 220 readings
  - average ambiguity for a corpus of newspaper texts: 78 with an average sentence length of 11.43 syntactic words (Gepard)
  - extreme case: $6.4875 \cdot 10^{22}$ for a single sentence (BLOCK 1995)

Stochastic models

- sources of ambiguity:
  - lexical ambiguity
  - attachment
    - *We saw the Eiffel Tower flying to Paris.*
  - coordination:
    - *old men and women*
  - NP segmentation
    - *. . . der Sohn des Meisters Geld*

Stochastic models

- example: PP-attachment
  the ball with the dots in the bag on the table
- grows exponentially (catalan) with the number of PPs

$$C(n) = \frac{1}{n+1} \binom{2n}{n}$$

# PPs	# parses
2	2
3	5
4	14
5	132
6	469
7	1430
8	4867

Stochastic models

- coverage
  - partial parser (WAUSCHKUHN 1996): 56.5% of the sentences
  - Gepard: 33.51%
  - on test suites (better lexical coverage, shorter and less ambiguous sentences) up to 66%

Stochastic models

- alternative: probabilistic context-free grammars (PCFG)
- estimation of derivation probabilities for all rules

$$\Pr(N \rightarrow \zeta)$$

or

$$\Pr(N \rightarrow \zeta | N) \quad \text{mit} \quad \sum_{\zeta} \Pr(N \rightarrow \zeta) = 1$$

- e.g.

S $\rightarrow$ NP VP	0.8
S $\rightarrow$ Aux NP VP	0.15
S $\rightarrow$ VP	0.05

Stochastic models

- language models: assigning a probability to a terminal string

$$\Pr(w_{1,n}) = \sum_{t_{1,n}} \Pr(t_{1,n})$$

(several derivations for a sentence)

$$= \sum_{t_{1,n}} \prod_{r_j \in t_{1,n}} \Pr(r_j)$$

- determining the most probable word form sequence

Stochastisches Basismodell

- disambiguation: determining of the most probable derivation

$$t_{1,n} = \arg\max_{t_{1,n} \in T} \Pr(t_{1,n})$$

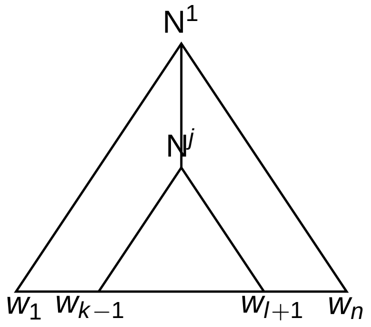
$$= \arg\max_{t_{1,n} \in T} \prod_{r_j \in t_{1,n}} \Pr(r_j)$$

Stochastic models

- independence assumption:

$$\Pr(N^j_{k,l} \rightarrow \zeta | N^1, \ldots., N^{j-1}, w_1, \ldots, w_{k-1}, w_{l+1}, \ldots, w_n)$$

$$= \Pr(N^j_{k,l} \rightarrow \zeta)$$



Stochastic models

- evaluation: PARSEVAL-metric (BLACK ET AL. 1991)
- comparison with a reference annotation (*gold standard*)
- labelled recall

$$LR = \frac{\text{\# correct constituents in the output}}{\text{\# constituents in the gold standard}}$$

- labelled precision

$$LP = \frac{\text{\# correct constituents in the output}}{\text{\# constituents in the output}}$$

Stochastic models

- crossing brackets
  a constituent of a parse tree contains parts of two constituents from the reference, but not the complete ones.

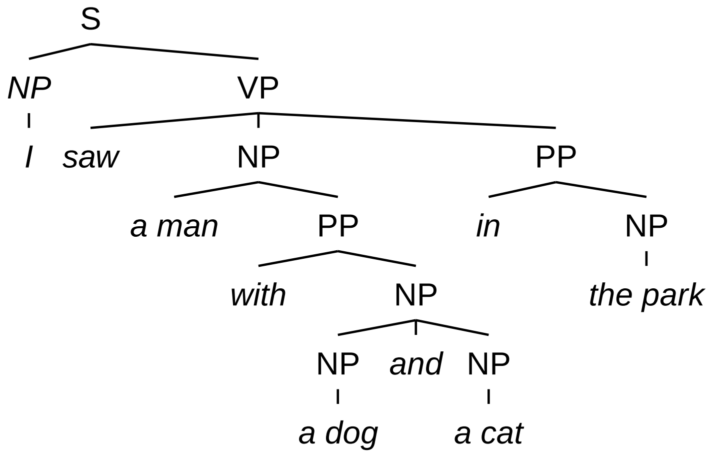
```
output:         [ [ A  B     C ] [ D  E ] ]
gold standard:  [ [ A  B ] [ C      D  E ] ]
```

$$CB = \frac{\text{\# crossing brackets}}{\text{\# sentences}}$$

$$0CB = \frac{\text{\# sentences without crossing brackets}}{\text{\# sentences}}$$

Stochastic models

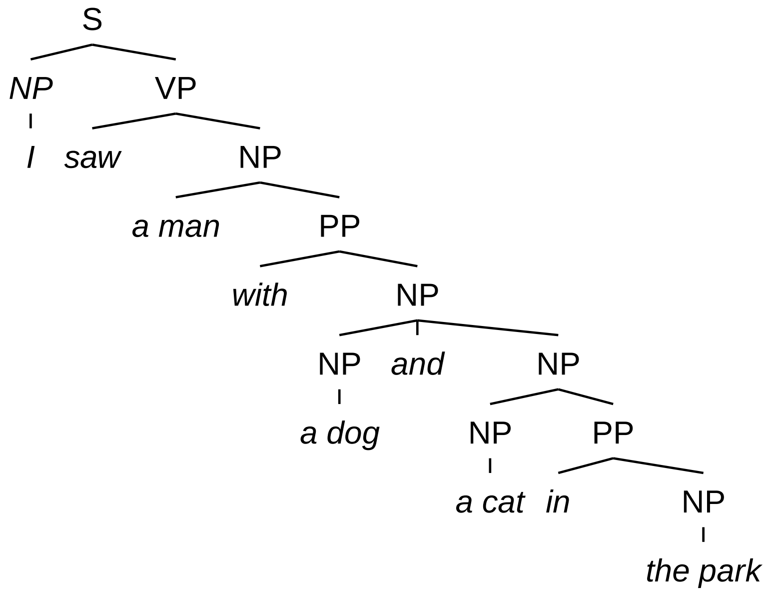
- How meaningful are the results?
- gold standard:



[I [saw [[a man] [with [[a dog] and [a cat]]]] [in [the park]]]]

Stochastic models

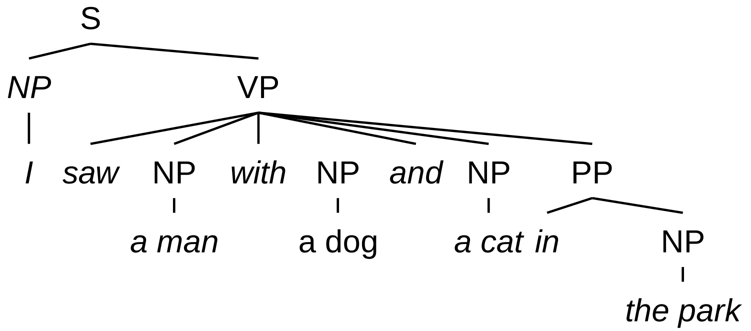
- 1st result: one erroneous attachment



[I [saw [[a man] [with [[a dog] and [[a cat] [in [the park]]]]]]]]

Stochastic models

- 2nd result: almost flat analysis
  - the parser tries to avoid any decisions on attachments



[I [saw [a man] with [a dog] and [a cat] [in [the park]]]]

Stochastic models

- 1st result
  [I [saw [[a man] [with [[a dog] and [[a cat] [in [the park]]]]]]]]
  [I [saw [[a man] [with [[a dog] and [a cat]]]] [in [the park]]]]
  $LR = \frac{7}{10} = 0.7 \qquad LP = \frac{7}{11} = 0.64 \qquad CB = \frac{3}{1} = 3$
- 2nd result
  [I [saw [a man] with [a dog] and [a cat] [in [the park]]]]
  [I [saw [[a man] [with [[a dog] and [a cat]]]] [in [the park]]]]
  $LR = \frac{7}{10} = 0.7 \qquad LP = \frac{7}{7} = 1 \qquad CB = \frac{0}{1} = 0$
- alternative (LIN 1996):
  transformation of the PS-tree into a dependency tree and evaluation of attachment errors

Stochastic models

- training: estimation of rule-application probabilities
- simplest case: treebank grammars
  (CHARNIAK 1996)

$$\Pr(N \rightarrow \zeta | N) = \frac{C(N \rightarrow \zeta)}{\sum_{\xi} C(N \rightarrow \xi)} = \frac{C(N \rightarrow \zeta)}{C(N)}$$

- Penn treebank: 10605 rules, among them 3943 only seen once
- results for sentences with up to 40 word forms:
  - LR = 80.4%, LP = 78.8%
  - constituents without crossing brackets: 87.7%

Stochastic models

- parsing with a modified EARLEY/CYK algorithm
- dynamic programming:
  - recursively constructing the parsing table and selecting the locally optimal interpretation

Stochastic models

- problem: independence assumption is systematically wrong
  - subject is more often pronominalized than the object
    - particularly in spoken language
    - consequence of the information structure
  - subcategorisation preferences disambiguate attachment problems
    - attachment to an NP is more frequent that attachment to the verb (2:1)
    - but: some verbs enforce an attachment of certain prepositions

  Moscow sent more than 100.000 soldiers into Afghanistan.

    - *send* requires a direction (*into*)
      $\rightarrow$ modelling of lexical dependencies becomes necessary

Stochastic models

- lexical dependencies cannot be expressed in a PCFG
  - only stochastic dependence on the dominating non-terminal

$$\Pr(N \rightarrow \zeta | N)$$

- extending the stochastic model with additional conditions

Stochastic models

- $\rightarrow$ lexicalised rule-application probabilities (CHARNIAK 2000)

$$\Pr(N \rightarrow \zeta | N, h(r))$$

- additionally considering the dependence (CHARNIAK 2000, COLLINS 1999)
  - on the head of the immediately dominating phrase level

$$\Pr(r = N \rightarrow \zeta | N, h(r), h(m(r)))$$

  - on the head of the two dominating phrase levels

$$\Pr(r = N \rightarrow \zeta | N, h(r), h(m(r)), h(m(m(r))))$$

Stochastic models

- problem: data sparseness
  - backoff
  - smoothing
  - stochastic modelling of the dependency of the sister nodes from the head as a Markov process (COLLINS 1999)

Stochastic models

- quality (CHARNIAK 2000)

sentence length $\leq$ 40					
parser	LR	LP	CB	0CB	2CB
COLLINS 1999	88.5	88.7	0.92	66.7	87.1
CHARNIAK 2000	90.1	90.1	0.74	70.1	89.6

sentence length $\leq$ 100					
parser	LR	LP	CB	0CB	2 CB
COLLINS 1999	88.1	88.3	1.06	64.0	85.1
CHARNIAK 2000	89.6	89.5	0.88	67.6	87.7

Stochastic models

- data orientierted parsing (DOP) (BOD 1992, 2003)
  - decomposition of the parse trees inro partial trees up to a depth of n ($n \leq 6$)
  - estimation of the frequency of all partial trees
  - determining the derivation probability for an output structure as the sum of all derivation possibilities
  - closed computation no longer possible $\rightarrow$ Monte-Carlo sampling
  - LR=90.7%, LP=90.8% (sentence length $\leq$ 100)

Stochastic models

- supertagging (BANGALORE 1997)
  - decomposition of the parse tree into lexicalised tree fragments
    - in analogy to a Tree Adjoining Grammar (TAG)
  - using the tree fragments as structurally rich lexical categories
  - training of a stochastic tagger
  - selection of the most probable sequence of tree fragments $\rightarrow$ almost parsing
  - reconstruction of a parse tree out of the tree fragments
  - better results (lower perplexity) with a Constraint Dependency Grammar (HARPER 2002)
    - even if trained on erroneous treebanks (HARPER 2003)

Stochastic models

- applications
  - approximative parsing for unrestricted text
    - information extraction
    - discourse analysis
  - analysis of ungrammatical input
  - language models for speech recognition
  - grammar induction

Restricted phrase-structure models

- linguistic goals:
  - define the rules of a grammar in a way that natural languages can be distinguished from artificial ones
  - specify general rule schemata which are valid for every language
    → X-bar schema (Jackendoff, 1977)
  - constraints on possible rule instances are principles of the grammar
    → universal grammar

Restricted phrase-structure models

- assumption: a phrase is always an extension of a lexical element

  VP → V NP
  reads the book
  NP → AP N
  dancing girls
  AP → PP A
  with reservations accepted
  PP → P NP
  with the children

  - there cannot be any rules of the type

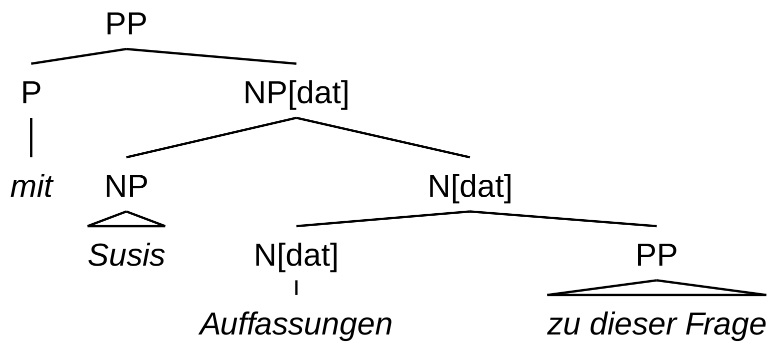
    NP → V AP
    VP → N PP
    ...

Restricted phrase-structure models

- two different kinds of categories
  - lexical element: head
  - phrasal elements: modifier
- head principle: Every phrase has exactly one head.
- phrase principle: Every non-head is a phrase

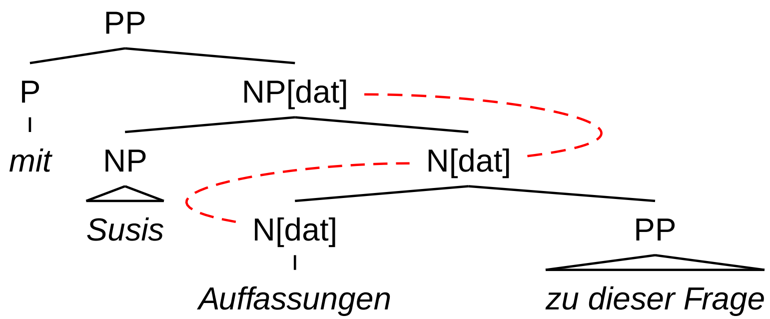
Restricted phrase-structure models

- head feature principle: The morphological (agreement-)features of a phrase are realized at its head



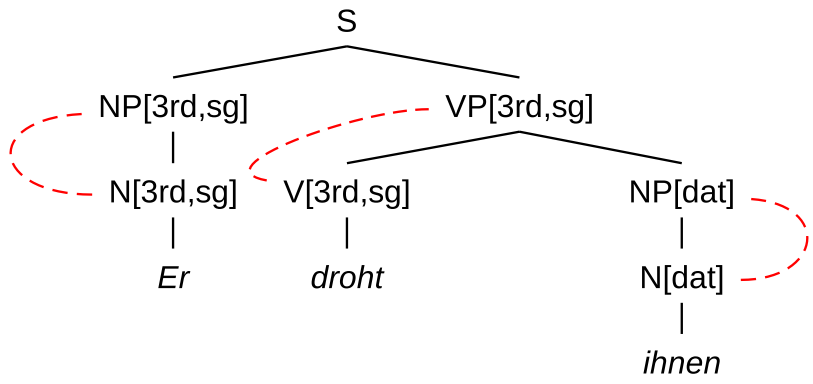
Restricted phrase-structure models

- projection line, head line: path from a complex category to its lexical head



Restricted phrase-structure models

- phrases are maximum projections of the head
  - case feature of a nominal head is only projected up to the NP level, not to the VP level
  - VP receives its agreement features from its head (the verb)



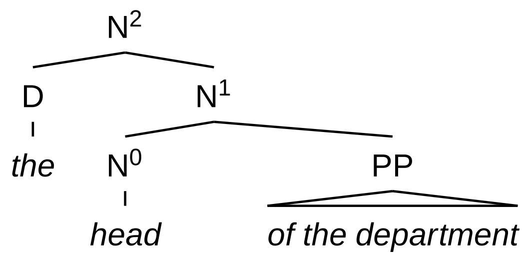
Restricted phrase-structure models

- complexity levels: NP has a higher (actually highest) complexity than N

  head
  head of the department
  head of the department who addressed the meeting

Restricted phrase-structure models

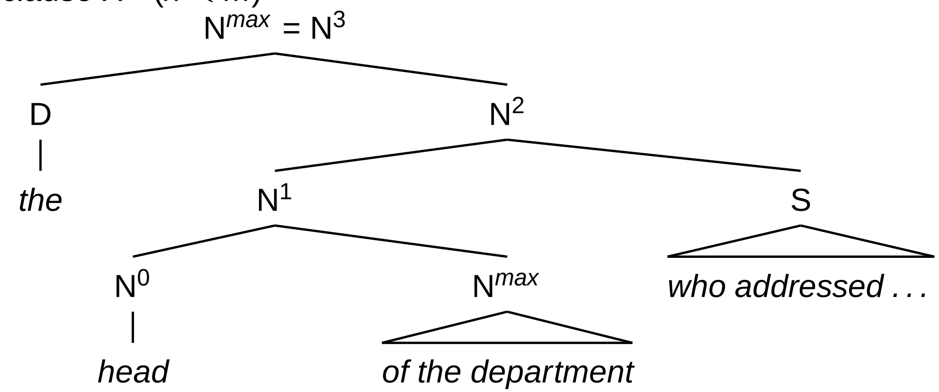
- level indicees to describe complexity levels (HARRIS 1951)
  - lexical level: X^0, head of the phrase
  - phrasal level: X^{max} or XP, phrases which cannot further be extended
  - $X \in \{N, V, A, P\}$



Restricted phrase-structure models

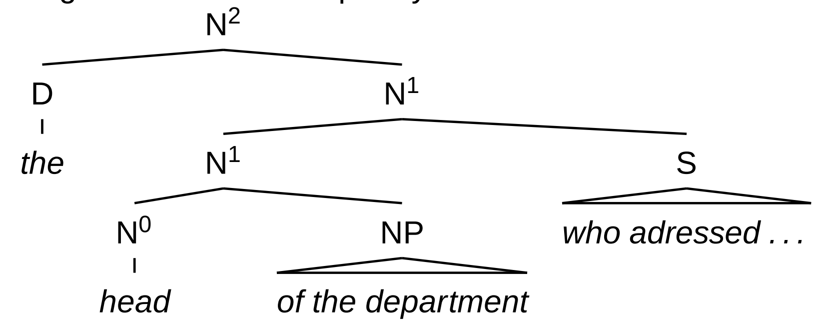
- observation:
  PP has a closer relationship to the head than a relative clause (cannot be exchanged without changing the attachment)
  the head of the department who addressed the meeting
  the head who addressed the meeting of the department

→ PPs belong to a lower complexity level X^n than the relative clause X^m ($n < m$)



Restricted phrase-structure models

- adjunction: constituents with the same distribution may get assigned the same complexity level



Restricted phrase-structure models

- three complexity level are sufficient
  - language specific parameter?
- rules:

  $NP \rightarrow D\ N^1$
  $N^1 \rightarrow N^1\ S$
  $N^1 \rightarrow N^0\ (NP)$

Restricted phrase-structure models

- adjunction for prepositional phrases

  $N^1 \rightarrow N^1\ PP$

  man with the glasses

- recursive application
  man with the glasses at the window
  man at the window with the glasses

- left NP-adjuncts

  $N^1 \rightarrow NP\ N^1$

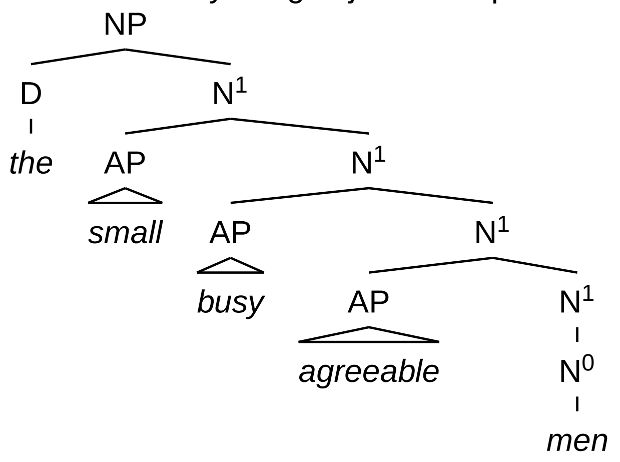
  a [Cambridge] [high quality] [middle class] student

Restricted phrase-structure models

- left adjective adjuncts

  $N^1 \rightarrow AP\ N^1$

- license "infinitely" long adjective sequences



Restricted phrase-structure models

- generalisation: Chomsky-adjunction

  $X^1 \rightarrow YP\ X^1$
  $X^1 \rightarrow X^1\ YP$

- schema for Chomsky-adjunction

X^i is the head

Restricted phrase-structure models

- level principle: The head of a category X^i is a category X^j, with $0 \leq j \leq i$.
  - the head has the same syntactic type as the constituent
  - the head is of lower structural complexity than the constituent

Restricted phrase-structure models

- X-bar schema: generalisation for arbitrary phrase structure rules:
- category variables

  $X \in \{V, N, P, A\}$

- category independence:

  Any categorial rules can be formulated using category variables.

Restricted phrase-structure models

- complement rule

  $X^1 \rightarrow YP^* \; X^0 \; YP^*$

- adjunct rule

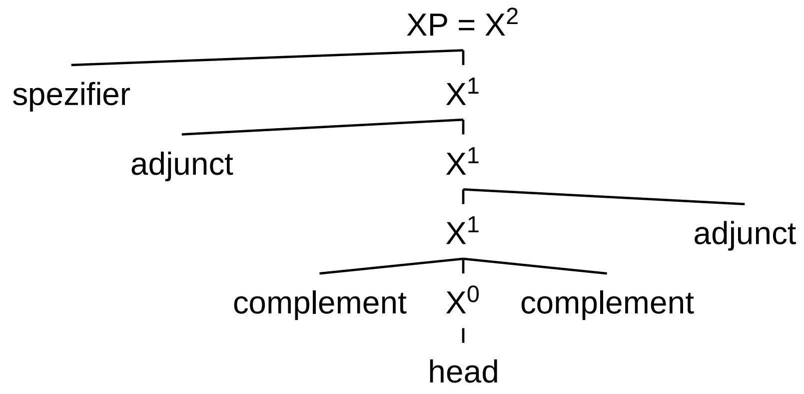
  $X^i \rightarrow YP^* \; X^i \; YP^*$ $\qquad 0 < i \leq max$

- specifier rule

  $X^{max} \rightarrow (YP) \; X^{max-1}$

Restricted phrase-structure models

- general schema for phrase structures with $max = 2$



Restricted phrase-structure models

- object restriction:

  subcategorized elements appear always at the transition between the X^0 and the X^1 level.
  - X^1 dominates immediately X^0 and the phrases subcategorized by X^0
- X-bar schema is order-free
- periphery of the head:

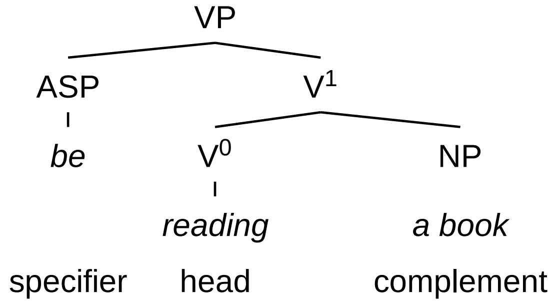
  The head of a projection is always peripheral.
- linearisation is a language specific parameter
- e.g. verb phrase
  - English: left peripheral
  - German: right peripheral

Restricted phrase-structure models

- X-bar schema is considered a constraint of universal grammar
  - restricts the set of possible phrase structure rules
  - gives a prognosis about all the acceptable structural descriptions for *all* natural languages

Restricted phrase-structure models

- example: English verb phrases



- aspectual auxiliary (progressive *be* and perfective *have*) as specifier (JACKENDOFF 1977)

Restricted phrase-structure models

- evidence for V^1
  - only V^1 can become topicalized, not VP

    They swore that John might have been taking heroin and

    . . . [$_{V^1}$ taking heroin] he might have been!
    *. . . * [$_{VP}$ been taking heroin] he might have!*
    *. . . * [$_{VP}$ have been taking heroin] he might!*

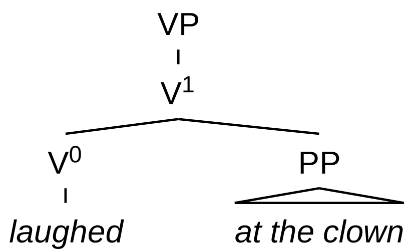
  - some verbs (e.g. *begin* or *see*) subcategorize V^1

    I saw John [$_{V^1}$ running down the road].
    ** I saw him [$_{VP}$ be running down the road].*
    ** I saw him [$_{VP}$ have finished his work].*

Restricted phrase-structure models

- structural distinction between complements and adjuncts
- complement:

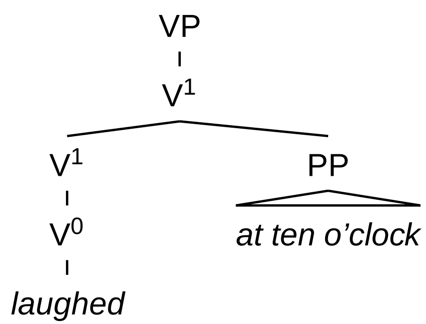
  He will work at the job.
  He laughed at the clown.



Restricted phrase-structure models

- adjunct:

  He will work at the office.
  He laughed at ten o'clock.

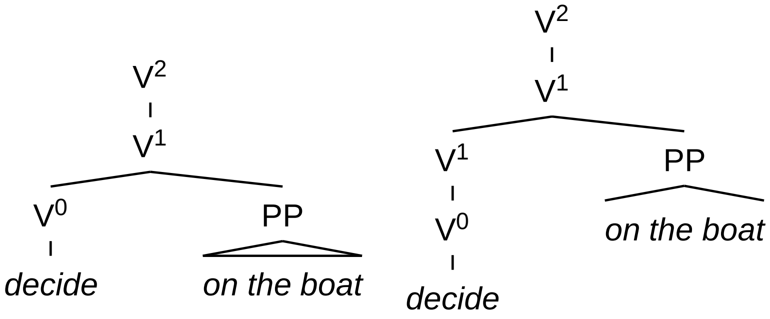


Restricted phrase-structure models

- evidence for the distinction between complements and adjuncts

1. structural ambiguity:

   He may decide on the boat.
   He couldn't explain last night.



Restricted phrase-structure models

2. passivization is possible for PP-complements, but not for PP-adjuncts

   [This job] needs to be worked at by an expert.
   ** [This office] is worked at by a lot of people.*

   [The clown] was laughed at by everyone.
   ** [Ten o'clock] was laughed at by everyone.*

3. when passivizing ambiguous constructions the adjunct reading disappears

   [The boat] was decided on after lengthy deliberation.
   [Last night] couldn't be explained by anyone.

   more evidence from phenomena like pronominalization, ordering restrictions, subcategorization, optionality and gapping in coordinated structures ...

Unification-based grammars

- feature structures
- rules with complex categories
- subcategorization
- movement

Feature structures

- feature structures describe linguistic objects (lexical items or phrases) as sets of attribute value pairs
- complex categories: name of the category may be part of the feature structure

  Haus: $\begin{bmatrix} \text{cat} & \text{N} \\ \text{case} & \text{nom} \\ \text{num} & \text{sg} \\ \text{gen} & \text{neutr} \end{bmatrix}$

- a feature structure is a functional mapping from a finite set of attributes to the set of possible values
  - unique names for attributes / unique value assignment
  - number of attributes is finite but arbitrary
  - feature structure can be extended by additional features

Feature structures

- partial descriptions: underspecified feature structures

Frauen: $\begin{bmatrix} \text{cat} & \text{N} \\ \text{num} & \text{pl} \\ \text{gen} & \text{fem} \end{bmatrix}$

Feature structures

- subsumtion:

  A feature structure M_1 subsumes a feature structure M_2 iff every attribute-value pair from M_1 is also contained in M_2.

  $\rightarrow$ not all pairs from M_2 need also be in M_1

- constraint-based notation (SHIEBER 1986): $M_1 \sqsubseteq M_2$
  - M_2 contains a superset of the constraints contained in M_1
  - M_2 is an extension of M_1 (POLLARD UND SAG 1987)
  - M_1 is less informative than M_2 (SHIEBER 1986, POLLARD UND SAG 1987)

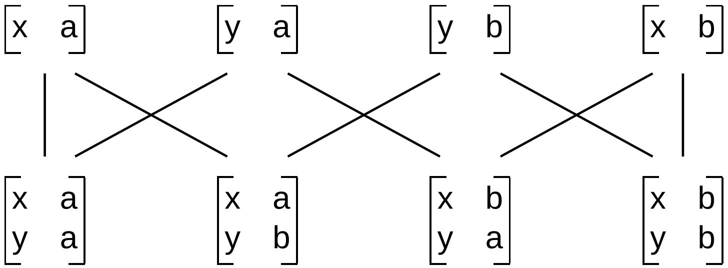
  but:
  - M_1 is more general than M_2

- alternative notation:

  instance-based (POLLARD UND SAG 1987): $M_1 \succeq M_2$

Feature structures

- subsumtion hierarchy



Feature structures

- formal properties of subsumtion
  - reflexive: $\forall M_i . M_i \sqsubseteq M_i$
  - transitive: $\forall M_i \forall M_j \forall M_k . M_i \sqsubseteq M_j \wedge M_j \sqsubseteq M_k \rightarrow M_i \sqsubseteq M_k$
  - antisymmetrical: $\forall M_i \forall M_j . M_i \sqsubseteq M_j \wedge M_j \sqsubseteq M_i \rightarrow M_i = M_j$
- subsumtion relation defines a partial order
- not all feature structures need to be in a subsumtion relation

Feature structures

- unification I (subsumtion-based)

  If M_1, M_2 and M_3 are feature structures, then M_3 is the unification of M_1 and M_2

  $$M_3 = M_1 \sqcup M_2$$

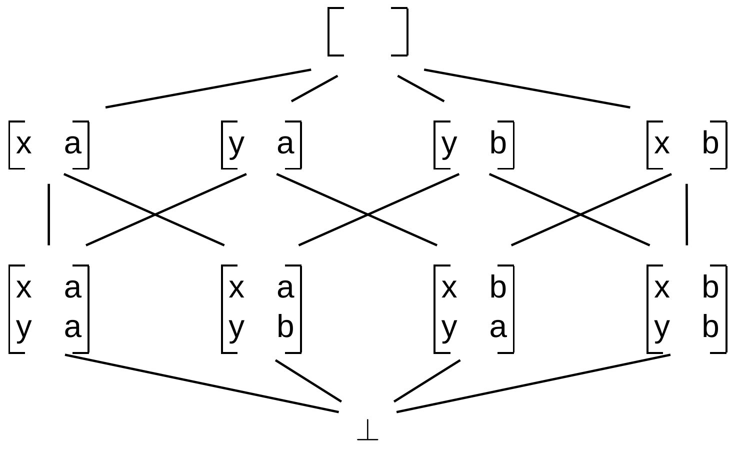
  iff
  - M_3 is subsumed by M_1 and M_2 and
  - M_3 subsumes all other feature structures, that are also subsumed by M_1 and M_2.
- result of a unification (M_3) is the most general feature structure which is subsumed by M_1 and M_2

Feature structures

- not all feature structures are in a subsumtion relation
  → unification may fail
- completing the subsumtion hierarchy to a lattice
  - bottom (⊥): inconsistent (overspecified) feature structure
  - top (⊤): totally underspecified feature structure corresponds to an unnamed variable ([ ])

Feature structures

- subsumtion lattice



Feature structures

- unification II (based on the propositional content) (Pollard und Sag 1987)

  The unification of two feature structures M_1 und M_2 is the conjunction of all propositions from the feature structures M_1 and M_2.
- unification combines two aspects:
  1. test of compatibility
  2. accumulation of information
- result of a unification combines two aspects
  1. Boolean value whether the unification was successful
  2. union of the compatible information from both feature structures

Feature structures

- formal properties of the unification
  - idempotent: $M \sqcup M = M$
  - commutative: $M_i \sqcup M_j = M_j \sqcup M_i$
  - associative: $(M_i \sqcup M_j) \sqcup M_k = M_i \sqcup (M_j \sqcup M_k)$
  - neutral element: $\top \sqcup M = M$
  - zero element: $\perp \sqcup M = \perp$
- unification and subsumtion can be mutally defined from each other

  $M_i \sqsubseteq M_j \leftrightarrow M_i \sqcup M_j = M_j$

Feature structures

- recursive feature structures: conditions are not to be defined for individual features but complete feature collections (data abstraction)
- value of an attribute is again a feature structure

$$\textit{Frauen}: \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 0 \\ \text{agr} & \begin{bmatrix} \text{num} & \text{pl} \\ \text{gen} & \text{fem} \end{bmatrix} \end{bmatrix}$$

Feature structures

- access to the values through paths

  $\langle$ cat $\rangle$ = N
  $\langle$ bar $\rangle$ = 0
  $\langle$ agr num $\rangle$ = pl
  $\langle$ agr gen $\rangle$ = fem

  $$\langle \text{agr} \rangle = \begin{bmatrix} \text{num} & \text{pl} \\ \text{gen} & \text{fem} \end{bmatrix}$$

Feature structures

- unification III (constructive algorithm)

  Two feature structures M_1 and M_2 unify, iff for every common feature of both structures
  - in case of atomic values both value assignments are identical or
  - in case of complex values both values unify.

  If successful unification produces as a result the set of all complete paths from M_1 *and* M_2 with their corresponding values. If unification fails the result will be $\perp$.

Feature structures

- recursive data structures can be used
  - lists
  - trees

$$(\text{A B C}) \quad \Longrightarrow \quad \begin{bmatrix} \text{first} & \text{A} \\ \text{rest} & \begin{bmatrix} \text{first} & \text{B} \\ \text{rest} & \begin{bmatrix} \text{first} & \text{C} \\ \text{rest} & \text{nil} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Feature structures

- example: subcategorisation list

$$(\text{NP[dat] NP[akk]}) \quad \Longrightarrow \quad \begin{bmatrix} \text{first} & \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{cas} & \text{dat} \end{bmatrix} \\ \text{rest} & \begin{bmatrix} \text{first} & \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{cas} & \text{akk} \end{bmatrix} \\ \text{rest} & \text{nil} \end{bmatrix} \end{bmatrix}$$

- two lists unify iff
  - they have the same length and
  - their elements unify pairwise.

Feature structures

- information in a feature structure is conjunctively combined
- feature structures might also contain disjunctions

$$\begin{bmatrix} \text{agr} & \left\{ \begin{bmatrix} \text{cas} & \text{nom} \\ \text{gen} & \text{masc} \\ \text{num} & \text{sg} \end{bmatrix} \begin{bmatrix} \text{cas} & \text{gen} \\ \text{gen} & \text{fem} \\ \text{num} & \text{sg} \end{bmatrix} \begin{bmatrix} \text{cas} & \text{dat} \\ \text{gen} & \text{fem} \\ \text{num} & \text{sg} \end{bmatrix} \begin{bmatrix} \text{cas} & \text{gen} \\ \text{num} & \text{pl} \end{bmatrix} \right\} \end{bmatrix}$$

Rules with complex categories

- categories with complexity level information

$$\begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{D} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 1 \end{bmatrix}$$

- modelling of government

$$\begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 0 \end{bmatrix} \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{cas} & \text{gen} \end{bmatrix}$$

Rules with complex categories

- representing the rule structure as a feature structure

  example: binary branching rule: $X0 \rightarrow X1\ X2$

$$\begin{bmatrix} X0 & \begin{bmatrix} cat & N \\ bar & 2 \end{bmatrix} \\ X1 & \begin{bmatrix} cat & D \\ bar & 0 \end{bmatrix} \\ X2 & \begin{bmatrix} cat & N \\ bar & 1 \end{bmatrix} \end{bmatrix}$$

Rules with complex categories

- representation of feature structures as path equations

$$\begin{bmatrix} X0 & \begin{bmatrix} cat & N \\ bar & 2 \end{bmatrix} \\ X1 & \begin{bmatrix} cat & D \\ bar & 0 \end{bmatrix} \\ X2 & \begin{bmatrix} cat & N \\ bar & 1 \end{bmatrix} \end{bmatrix} \Longrightarrow \begin{array}{l} \langle\ XO\ cat\ \rangle = N \\ \langle\ XO\ bar\ \rangle = 2 \\ \langle\ X1\ cat\ \rangle = D \\ \langle\ X1\ bar\ \rangle = 0 \\ \langle\ X2\ cat\ \rangle = N \\ \langle\ X2\ bar\ \rangle = 1 \end{array}$$

- features may corefer (coreference, reentrancy, structure sharing)

Rules with complex categories

- applications of coreference:
  - agreement: $\langle$ X1 agr $\rangle$ = $\langle$ X2 agr $\rangle$
  - projection: $\langle$ X0 agr $\rangle$ = $\langle$ X2 agr $\rangle$

Rules with complex categories

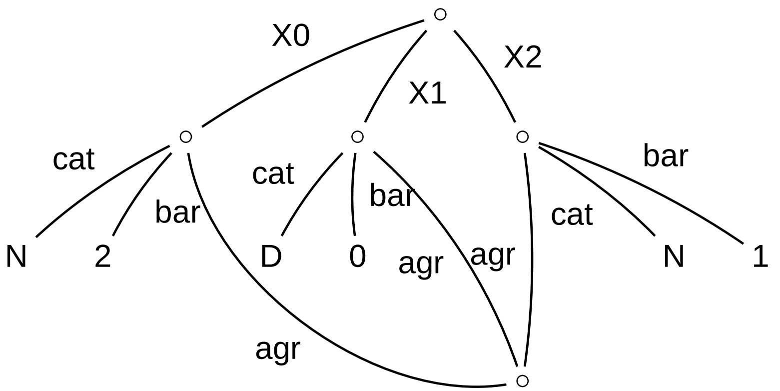
- representation in feature matricees by means of coreference marker or path equations

$$\begin{bmatrix} X0 & \begin{bmatrix} cat & N \\ bar & 2 \\ agr & \boxed{1} \end{bmatrix} \\ X1 & \begin{bmatrix} cat & D \\ bar & 0 \\ agr & \boxed{1} \end{bmatrix} \\ X2 & \begin{bmatrix} cat & N \\ bar & 1 \\ agr & \boxed{1} \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} X0 & \begin{bmatrix} cat & N \\ bar & 2 \\ agr & \end{bmatrix} \\ X1 & \begin{bmatrix} cat & D \\ bar & 0 \\ agr & = \langle\ X0\ agr\ \rangle \end{bmatrix} \\ X2 & \begin{bmatrix} cat & N \\ bar & 1 \\ agr & = \langle\ X0\ agr\ \rangle \end{bmatrix} \end{bmatrix}$$

- coreference corresponds to a named variable

Rules with complex categories

- feature structures with coreference correspond to a directed acyclic graph



Rules with complex categories

- generalised adjunct rule for prepositional phrases

$$\begin{bmatrix} \text{X0} & \begin{bmatrix} \text{cat} & \boxed{1} \\ \text{bar} & 1 \end{bmatrix} \\ \text{X1} & \begin{bmatrix} \text{cat} & \boxed{1} \\ \text{bar} & 1 \end{bmatrix} \\ \text{X2} & \begin{bmatrix} \text{cat} & \text{P} \\ \text{bar} & 2 \end{bmatrix} \end{bmatrix}$$

Rules with complex categories

- consequences of coreference on the information content:
  - structural equality (type identity): $\begin{bmatrix} \text{x} & [\,] \\ \text{y} & [\,] \end{bmatrix}$
  - referential identity (token identity): $\begin{bmatrix} \text{x} & \boxed{1}\,[\,] \\ \text{y} & \boxed{1} \end{bmatrix}$
  - a coreference is an additional constraint
  - equality is more general than identity: $\begin{bmatrix} \text{x} & [\,] \\ \text{y} & [\,] \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{x} & \boxed{1}\,[\,] \\ \text{y} & \boxed{1} \end{bmatrix}$
- definition of unification is not affected by the introduction of coreference

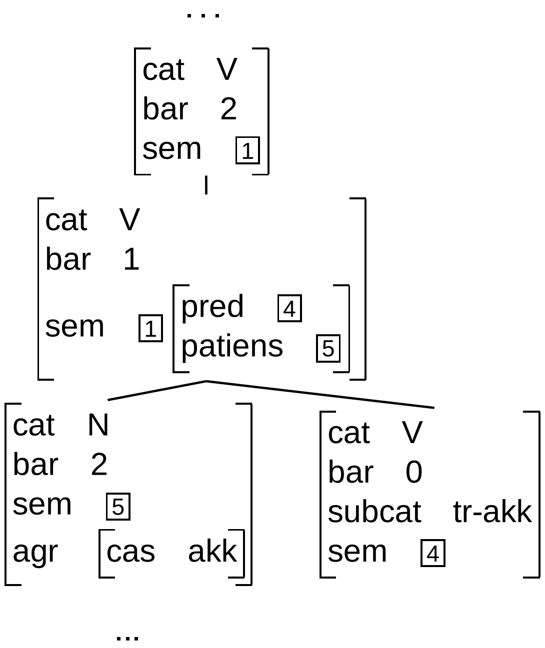
Rules with complex categories

- construction of arbitrary structural descriptions e.g. logical form

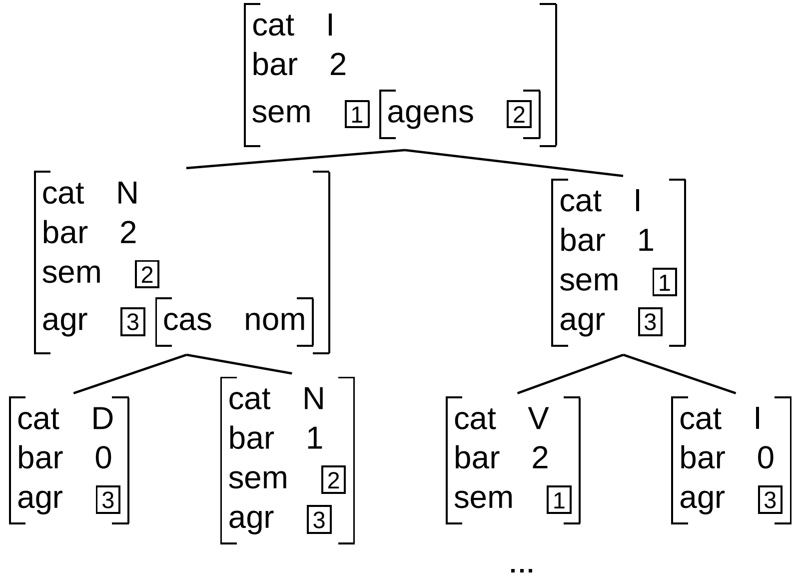
$$\begin{bmatrix} \text{cat} & \text{I} \\ \text{bar} & 2 \\ \text{sem} & \boxed{1}\begin{bmatrix} \text{agens} & \boxed{2} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{sem} & \boxed{2} \\ \text{agr} & \boxed{3}\begin{bmatrix} \text{cas} & \text{nom} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{I} \\ \text{bar} & 1 \\ \text{sem} & \boxed{1} \\ \text{agr} & \boxed{3} \end{bmatrix}$$

$$\begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 1 \\ \text{sem} & \begin{bmatrix} \text{pred} & \boxed{1} \\ \text{patiens} & \boxed{2} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{sem} & \boxed{2} \\ \text{agr} & \begin{bmatrix} \text{cas} & \text{akk} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 0 \\ \text{subcat} & \text{tr-akk} \\ \text{sem} & \boxed{1} \end{bmatrix}$$

Rules with complex categories



Rules with complex categories



Rules with complex categories

- construction of left recursive structures with right recursive rules
- left recursive rules (DCG-notation)

```
np(np(Snp,Spp)) --> np(Snp), pp(Spp).
np(np(Sd,Sn)) --> d(Sd), n(Sn).
```

- right recursive rules

```
np(np(Sd,Sn)) --> d(Sd), n(Sn).
np(Spps) --> d(Sd), n(Sn), pps(np(Sd,Sn),Spps).

pps(Snp,np(Snp,Spp)) --> pp(Spp).
pps(Snp,Spps) --> pp(Spp), pps(np(Snp,Spp),Spps).
```


Rules with complex categories

- example: *the house behind the street with the red roof*

```
?- np(S,[t,h,bts,wtrr],[ ]).
   np(Spps1) --> d(Sd), n(Sn), pps(np(Sd,Sn),Spps1).        S=Spps1
   . . .
   ?- pps(np(d(t),n(h)),Spps1,[bts,wtrr],Z1).
      pps(Snp2,Spps2) --> pp(Spp), pps(np(Snp,Spp),Spps2). Spps1=Spps2
      . . .
      ?- pps(np(np(d(t),n(h)),pp(bts)),Spps2,[wtrr],Z2)
         pps(Snp,np(Snp,Spp)) --> pp(Spp).

Snp = np(np(d([t]),n([h])),pp([bts])),
Spps2 = np(np(np(d([t]),n([h])),pp([bts])),pp([wtrr])
```

Rules with complex categories

- parsing with complex categories
  - test for identity has to be replaced by unifiability
  - but: unification is destructive
    - information is added to rules or lexical entries
    - feature structures need to be copied prior to unification

Subcategorization

- modelling of valence requirements as a list

$$\textit{geben:}\quad \begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 0 \\ \text{subcat} & \begin{bmatrix} \text{first} & \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{agr}|\text{cas} & \text{akk} \end{bmatrix} \\ \text{rest} & \begin{bmatrix} \text{first} & \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{agr}|\text{cas} & \text{dat} \end{bmatrix} \\ \text{rest} & \text{nil} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Subcategorisation

- processing of the information by means of suitable rules

$$\begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 0 \\ \text{subcat} & \boxed{1} \end{bmatrix} \rightarrow \boxed{2}\begin{bmatrix} \; \end{bmatrix} \begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 0 \\ \text{subcat} & \begin{bmatrix} \text{first} & \boxed{2} \\ \text{rest} & \boxed{1} \end{bmatrix} \end{bmatrix} \qquad \text{rule 1}$$

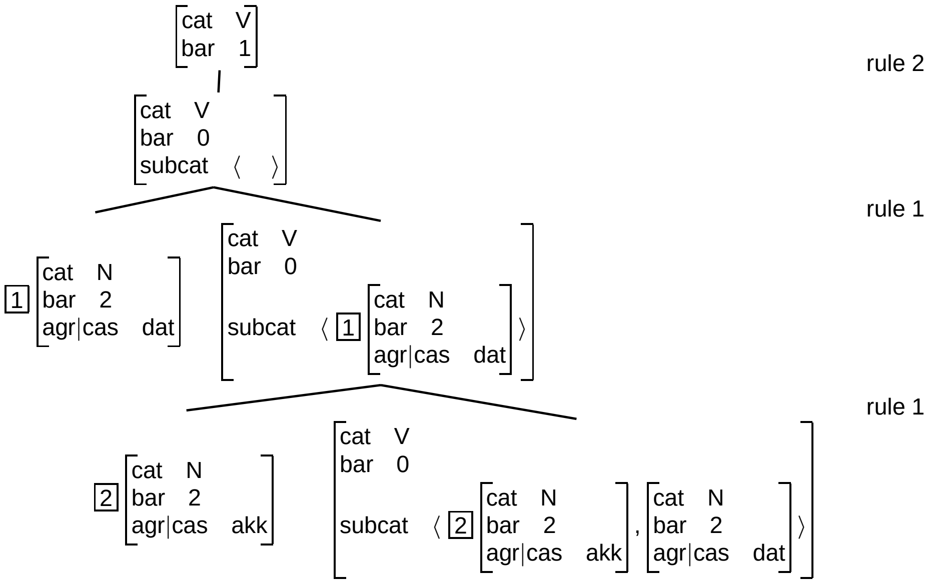
$$\begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 0 \\ \text{subcat} & \text{nil} \end{bmatrix} \qquad \text{rule 2}$$

Subcategorisation

- list notation

$$\textit{geben:}\quad \begin{bmatrix} \text{cat} & \text{V} \\ \text{bar} & 0 \\ \text{subcat} & \left\langle \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{agr}|\text{cas} & \text{akk} \end{bmatrix}, \begin{bmatrix} \text{cat} & \text{N} \\ \text{bar} & 2 \\ \text{agr}|\text{cas} & \text{dat} \end{bmatrix} \right\rangle \end{bmatrix}$$

Subcategorisation



Movement

- movement operations are unidirectional and procedural
- goal: declarative integration into feature structures
- slash operator

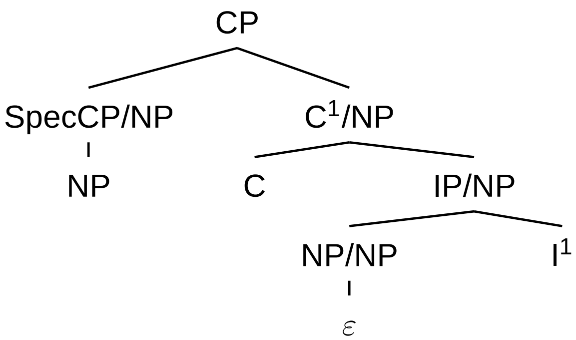
S/NP	sentence without a noun phrase
VP/V	verb phrase without a verb
S/NP/NP	
...	

  - first used in categorial grammar (BAR-HILLEL 1963)
  - also order sensitive variant: S\NP/NP

Movement

- topicalisation

CP → SpecCP/NP C^1/NP	
SpecCP/NP → NP	slash introduction
C^1/NP → C IP/NP	slash transition
IP/NP → NP/NP I^1	slash transition
NP/NP → ε	slash elimination



Movement

- encoding in feature structures: slash feature
  - moved constituents are connected to their trace by means of coreference
  - computation of the logical form is invariant against movement operations

Constraint-based models

- head-driven phrase-structure grammar (HPSG, Pollard and Sag 1987, 1994)
- inspired by the principles & parameter model of Chomsky (1981)
- constraints: implications over feature structures:
  if the premise can be unified with a feature structure unify the consequence with that structure.

$$\begin{bmatrix} type_1 \\ \ \end{bmatrix} \rightarrow \begin{bmatrix} X1|\ldots|\,XN & \boxed{1} \\ Y1|\ldots|YM & \boxed{1} \end{bmatrix}$$

Constraint-based models

- feature structures need to be typed *Haus:*

$$\begin{bmatrix} \textit{nomen} & \\ \text{cat} & \text{N} \\ \text{agr} & \begin{bmatrix} \textit{agr} & \\ \text{case} & \text{nom} \\ \text{num} & \text{sg} \\ \text{gen} & \text{neutr} \end{bmatrix} \end{bmatrix}$$

- extention of unification and subsumtion to typed feature structures
  - subsumtion:

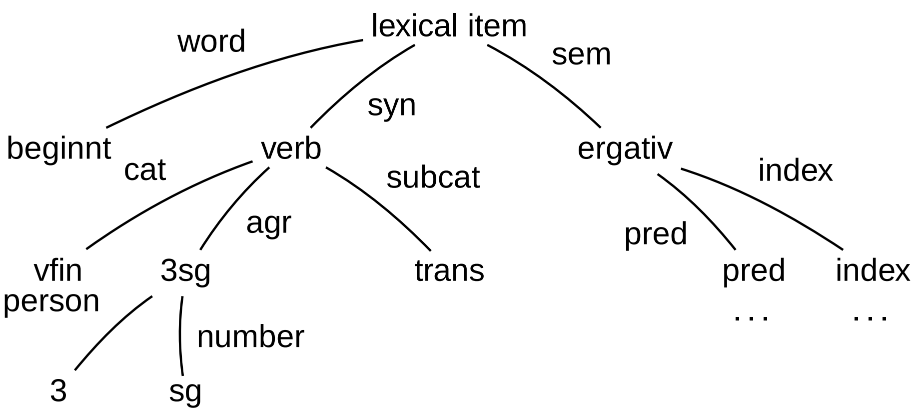
    $M_i^m \sqsubseteq M_j^n$ gdw. $M_i \sqsubseteq M_j$ und $m = n$

  - unification:

    $M_i^m \sqcup M_j^n = M_k^o$ gdw. $M_k = M_i \sqcup M_j$ und $m = n = o$

Constraint-based models

- graphical interpretation: types as node annotations



Constraint-based models

- types are organized in a type hierarchy:
  - partial order for types:

    sub(*verb*,*finite*)
    sub(*verb*,*finite*)
    ...
  - hierarchical abstraction
- subsumtion for types:

$$m \sqsubseteq n \quad \text{iff} \quad \begin{cases} \text{sub}(m, n) \\ \text{sub}(m, x) \wedge \text{sub}(x, n) \end{cases}$$

- unification for types:

$$m \sqcup n = o \quad \text{iff} \quad \begin{array}{l} m \sqsubseteq o \wedge n \sqsubseteq o \quad \text{and} \\ \neg\exists x . m \sqsubseteq x \wedge n \sqsubseteq x \wedge x \sqsubseteq o \end{array}$$

Constraint-based models

- subsumtion for typed feature structures:

$$M_i^m \sqsubseteq M_j^n \quad \text{iff} \quad \begin{array}{ll} M_i \sqsubseteq M_j & \text{and} \\ m \sqsubseteq n & \end{array}$$

- unification for typed feature structures:

$$M_i^m \sqcup M_j^n = M_k^o \quad \text{iff} \quad \begin{array}{ll} M_k = M_i \sqcup M_j & \text{and} \\ o = m \sqcup n & \end{array}$$

Constraint-based models

- HPSG: lexical signs

$$\left[\begin{array}{ll}
\textit{word} & \\
\text{PHON} & \\
\text{SYNSEM} & \left[\begin{array}{ll}
\textit{synsem} & \\
\text{LOC} & \left[\begin{array}{ll}
\textit{local} & \\
\text{CAT} & \left[\begin{array}{l} \textit{cat} \\ \text{HEAD} \\ \text{SUBCAT} \end{array}\right] \\
\text{CONT} & \left[\begin{array}{l} \textit{npro/ppro} \\ \text{INDEX} \\ \text{RESTR} \end{array}\right] \\
\text{CONX} & \left[\begin{array}{ll} \text{BACKGR} & \left\{ \left[\textit{psoa}\right], \ldots \right\} \end{array}\right]
\end{array}\right] \\
\text{NONLOC} &
\end{array}\right]
\end{array}\right]$$

Constraint-based models

- HPSG: phrasal signs
  - signs of type *phrase*
    additional features: Daughters, (Quantifier-Store)
  - most important special case:
    head-comp-struc

Constraint-based models

- DAUGHTERS (DTRS)
  - constituent structure of a phrase
  - HEAD-DTR (*phrase*)
  - COMP-DTRS (list of elementes of type *phrase*)

$$\left[\begin{array}{ll}
\textit{phrase} & \\
\text{PHON} & \langle \textit{Kim, walks} \rangle \\
\text{SYNSEM} & \text{S}[\textit{fin}] \\
\text{DTRS} & \left[\begin{array}{ll}
\textit{head-comp-struc} & \\
\text{HEAD-DTR} & \left[\begin{array}{ll} \textit{phrase} & \\ \text{PHON} & \langle \textit{walks} \rangle \\ \text{SYNSEM} & \text{VP}[\textit{fin}] \end{array}\right] \\
\text{COMP-DTRS} & \left\langle \left[\begin{array}{ll} \textit{phrase} & \\ \text{PHON} & \langle \textit{Kim} \rangle \\ \text{SYNSEM} & \text{NP}[\textit{nom}] \end{array}\right] \right\rangle
\end{array}\right]
\end{array}\right]$$

Constraint-based models

- head-feature principle
  - projection of head features to the phrase level
  - the HEAD-feature of a head structure corefers with the HEAD-feature of its head daughter.

$$\begin{bmatrix}\text{DTRS} & \begin{bmatrix}\textit{head-struc}\end{bmatrix}\end{bmatrix} \rightarrow \begin{bmatrix}\text{SYNSEM|LOC|CAT|HEAD} & \boxed{1}\\ \text{DTRS|HEAD-DTR|SYNSEM|LOC|CAT|HEAD} & \boxed{1}\end{bmatrix}$$

Constraint-based models

- subcategorisation principle
  - SUBCAT-list is ordered: relative obliqueness
  - subject is not structurally determinined, and therefore the element of the SUBCAT-list with the lowest obliqueness
  - obliqueness hierarchie
    - subject, primary object, secondary object, oblique prepositional phrases, verb complements, . . .
  - oblique subcategorisation requirements are bound first in the syntax tree

Constraint-based models

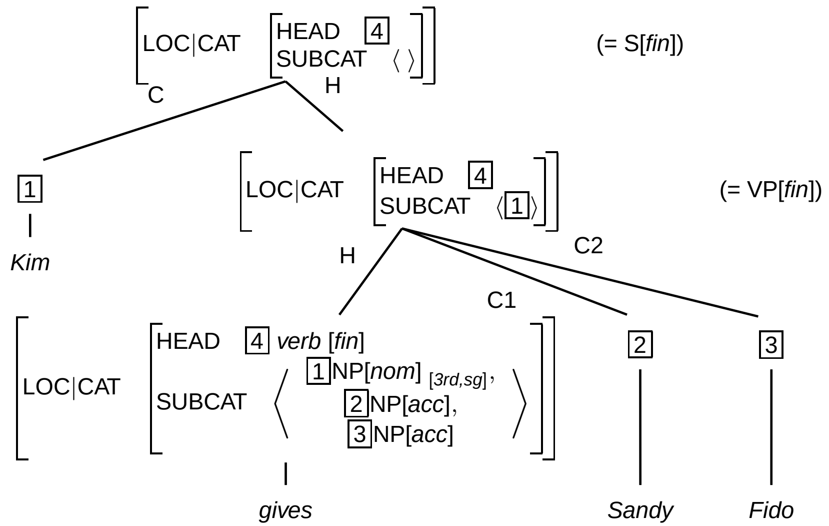
- subcategorisation principle:

  In a head-complement-phrase the SUBCAT-value of the head daughter is equal to the combination of the SUBCAT-list of the phrase with the SYNSEM-values of the complement daughters (arranged according to increasing obliqueness).

$$\begin{bmatrix}\text{DTRS} & \begin{bmatrix}\textit{head-compl-struc}\end{bmatrix}\end{bmatrix} \rightarrow \begin{bmatrix}\text{SYNSEM|LOC|CAT|SUBCAT} & \boxed{1}\\ \text{DTRS} & \begin{bmatrix}\text{HEAD-DTR|SYNSEM|LOC|CAT|SUBCAT} & \text{append}(\boxed{1},\boxed{2})\\ \text{COMP-DTRS} & \boxed{2}\end{bmatrix}\end{bmatrix}$$

Constraint-based models

- subcategorisation principle:



Constraint-based models

- more constraints for deriving a semantic description (predicate-argument structure, quantor handling, ...)
- advantages of principle-based modelling:
  - modularization: general requirements (e.g. agreement, construction of a semantic representation) are implemented once and not repeatedly in various rules
  - object-oriented modelling: heavy use of inheritance
  - context-free backbone of the grammar is removed almost completely; only very few general structural schemata remain (head-complement structure, head-adjunct structure, coordinated structure, ...)
  - integrated treatment of semantics in a general form

Questions to ask ...

... when defining a research project:

- What’s the problem?
- Which kind of linguistic/extra-linguistic knowledge is needed to solve ist?
- Which models and algorithms are available?
- Are their similar solutions for other / similar language?
- Which information can they capture and why?
- What are their computational properties?
- Can a model be applied directly or need it be modified?
- Which resources are necessary and need to be developed? How expensive this might be?
- Which experiments should be carried out to study the behaviour of the solution in detail?
- ...