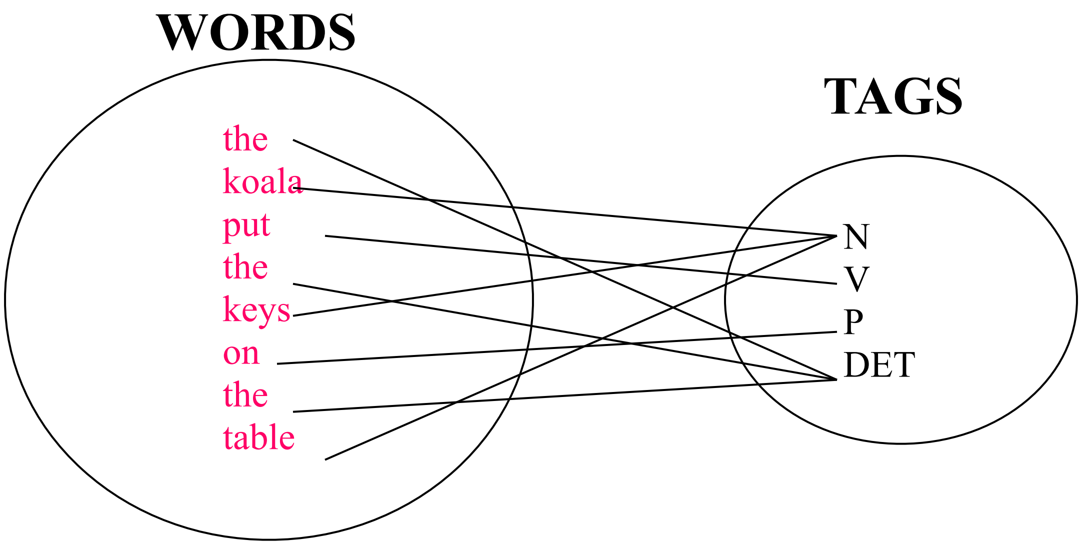
Word Classes and Part-of-Speech (POS) Tagging

Based on slides from Julia Hirschberg - www.cs.columbia.edu/~julia

Defining POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a corpus:



Applications for POS Tagging

- Speech synthesis pronunciation
  - *Lead* *Lead*
  - *INsult* *inSULT*
  - *OBject* *obJECT*
  - *OVERflow* *overFLOW*
  - *DIScount* *disCOUNT*
  - *CONtent* *conTENT*
- Parsing: e.g. *Time flies like an arrow*
  - Is *flies* an N or V?
- Word prediction in speech recognition
  - Possessive pronouns (*my, your, her*) are likely to be followed by nouns
  - Personal pronouns (*I, you, he*) are likely to be followed by verbs
- Machine Translation

Tag Ambiguity

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is ***to determine the POS tag for a particular instance of a word***

Tagging Whole Sentences with POS is Hard

- Ambiguous POS contexts
  - E.g., Time flies like an arrow.
- Possible POS assignments
  - Time/[V,N] flies/[V,N] like/[V,Prep] an/Det arrow/N
  - Time/N flies/V like/Prep an/Det arrow/N
  - Time/V flies/N like/Prep an/Det arrow/N
  - Time/N flies/N like/V an/Det arrow/N
  - …..

Some Ways to do POS Tagging

- Rule-based tagging
  - E.g. EnCG ENGTWOL tagger
- Transformation-based tagging
  - Learned rules (statistical and linguistic)
  - E.g., Brill tagger
- Stochastic, or, Probabilistic tagging
  - HMM (Hidden Markov Model) tagging

POS Tagging as a Sequence ID Task

- Given a sentence (a sequence of words, or observations)
  - Secretariat is expected to race tomorrow
- What is the best ***sequence of tags*** which corresponds to this ***sequence of observations***?
- Bayesian approach:
  - Consider all possible sequences of tags
  - Choose the tag sequence $t_1 \ldots t_n$ which is most probable given the observation sequence of words $w_1 \ldots w_n$

POS Tagging Equation

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

Now we have two probabilities to calculate:

•Probability of a word occurring given its tag

•Probability of a tag occurring given a previous tag

•We can calculate each of these from a POS-tagged corpus

Tag Transition Probabilities $P(t_i|t_{i-1})$

- Determiners likely to precede adjs and nouns but unlikely to follow adjs
  - The/DT red/JJ hat/NN;*Red/JJ the/DT hat/NN
  - So we expect P(NN|DT) and P(JJ|DT) to be high but P(DT|JJ) to be low
- Compute P(NN|DT) by counting in a tagged corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Word Likelihood Probabilities $P(w_i|t_i)$

- VBZ (3sg pres verb) likely to be is
- Compute P(is|VBZ) by counting in a tagged corpus:

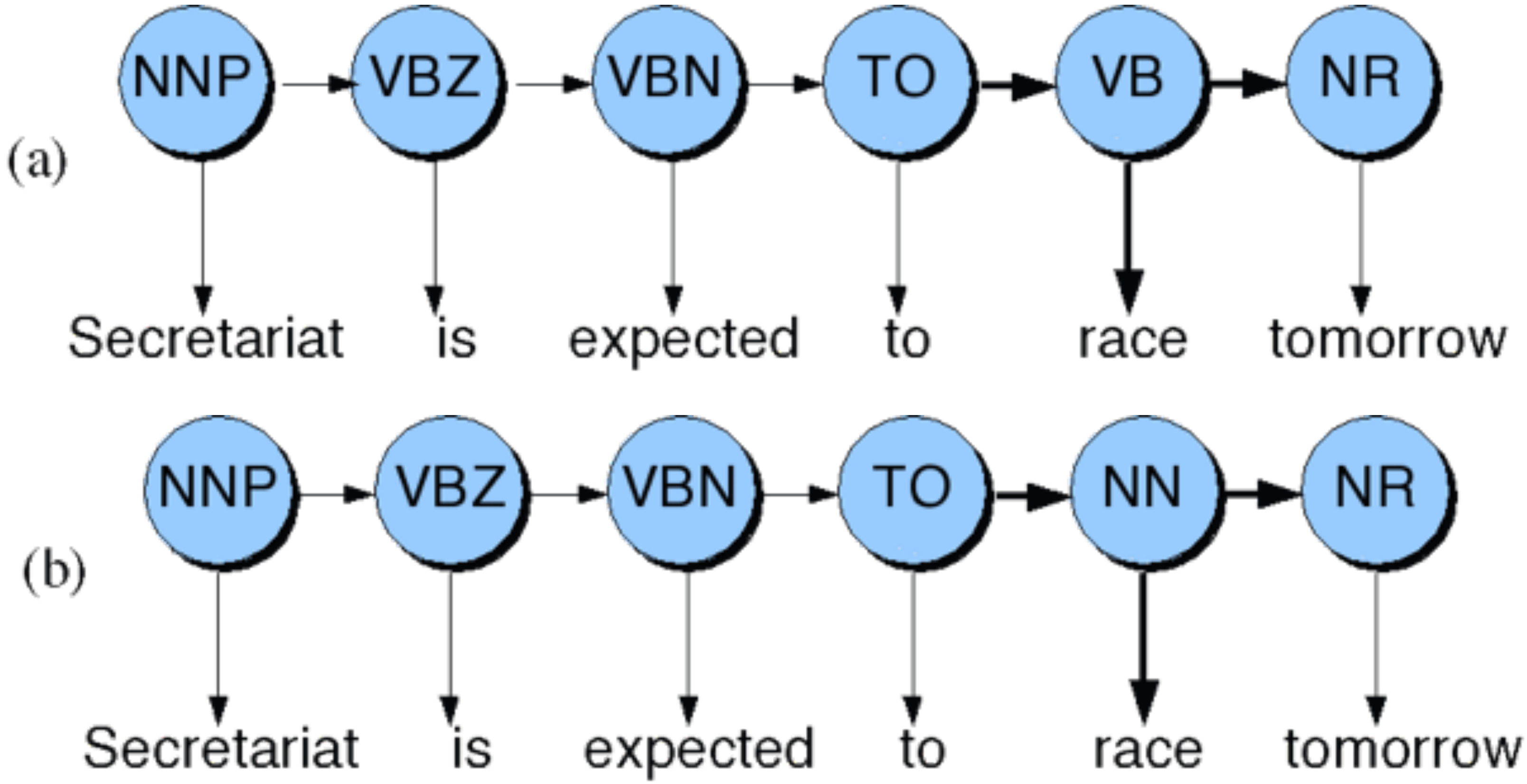
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

Some Data on race

- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN
- How do we pick the right tag for race in new data?

Disambiguating to race tomorrow



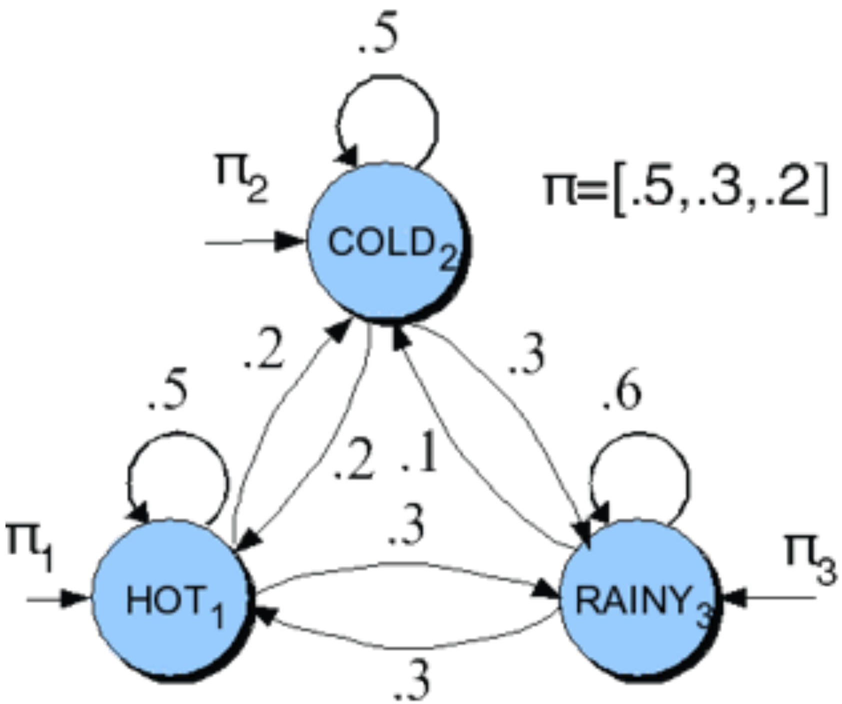
$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

- P(NN|TO) = .00047
- P(VB|TO) = .83
- P(race|NN) = .00057
- P(race|VB) = .00012
- P(NR|VB) = .0027
- P(NR|NN) = .0012
- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the verb reading

Markov Chains

- Markov Chains
  - Have transition probabilities like $P(t_i|t_{i-1})$
- A special case of weighted FSTs (FSTs which have probabilities or weights on the arcs)
- Can only represent unambiguous phenomena

A Weather Example: cold, hot, hot



Weather Markov Chain

- What is the probability of 4 consecutive rainy days?
- Sequence is rainy-rainy-rainy-rainy
- I.e., state sequence is 3-3-3-3
- P(3,3,3,3) =
  - $\pi_3 a_{33} a_{33} a_{33} a_{33} = 0.2 \times (0.6)^3 = 0.0432$

Weather Markov Chain

- What is the probability of 4 consecutive cold days?
- Sequence is cold-cold-cold-cold
- I.e., state sequence is 2-2-2-2
- P(2,2,2,2) =
  - $\pi_2 a_{22} a_{22} a_{22} a_{22} = 0.3 \times (0.5)^3 = 0.0375$

Markov Chain Defined

- A set of states $Q = q_1, q_2 \ldots q_N$
- Transition probabilities
  - A set of probabilities $A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$$

$$\sum_{j=1}^{N} a_{ij} = 1; \quad 1 \le i \le N$$

- Distinguished start and end states q_0, q_F

- Can have special initial probability vector π instead of start state

$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

  - An initial distribution over probability of start states
  - Must sum to 1

$$\sum_{j=1}^{N} \pi_j = 1$$

Hidden Markov Models

- Markov Chains are useful for representing problems in which
  - Observable events
  - Sequences to be labeled are unambiguous
- Problems like POS tagging are neither
- HMMs are useful for computing events we ***cannot*** directly observe in the world, using other events we ***can*** observe
  - Unobservable (Hidden): e.g., POS tags
  - Observable: e.g., words
  - We have to ***learn*** the relationships

Hidden Markov Models

- A set of states $Q = q_1, q_2 \ldots q_N$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$

$$\sum_{j=1}^{N} a_{ij} = 1; \quad 1 \le i \le N$$

- Observations $O = o_1, o_2 \ldots o_{N;}$
  - Each a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
- Observation likelihoods or emission probabilities
  - Output probability matrix $B = \{b_i(o_t)\}$

- Special initial probability vector π

$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

- A set of legal accepting states $QA \subset Q$

First-Order HMM Assumptions

- **Markov assumption: probability of a state depends only on the state that precedes it**

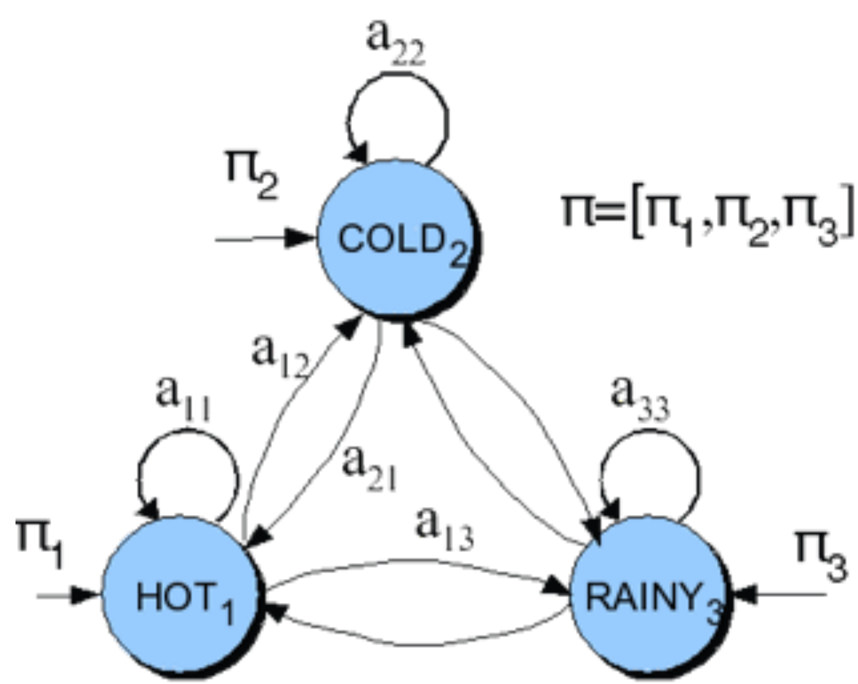
$$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

  - This is the same Markov assumption we made when we decided to represent sentence probabilities as the product of bigram probabilities

- **Output-independence assumption: probability of an output observation depends only on the state that produced the observation**

$$P(o_t \mid O_1^{t-1}, q_1^t) = P(o_t \mid q_t)$$

Weather Again



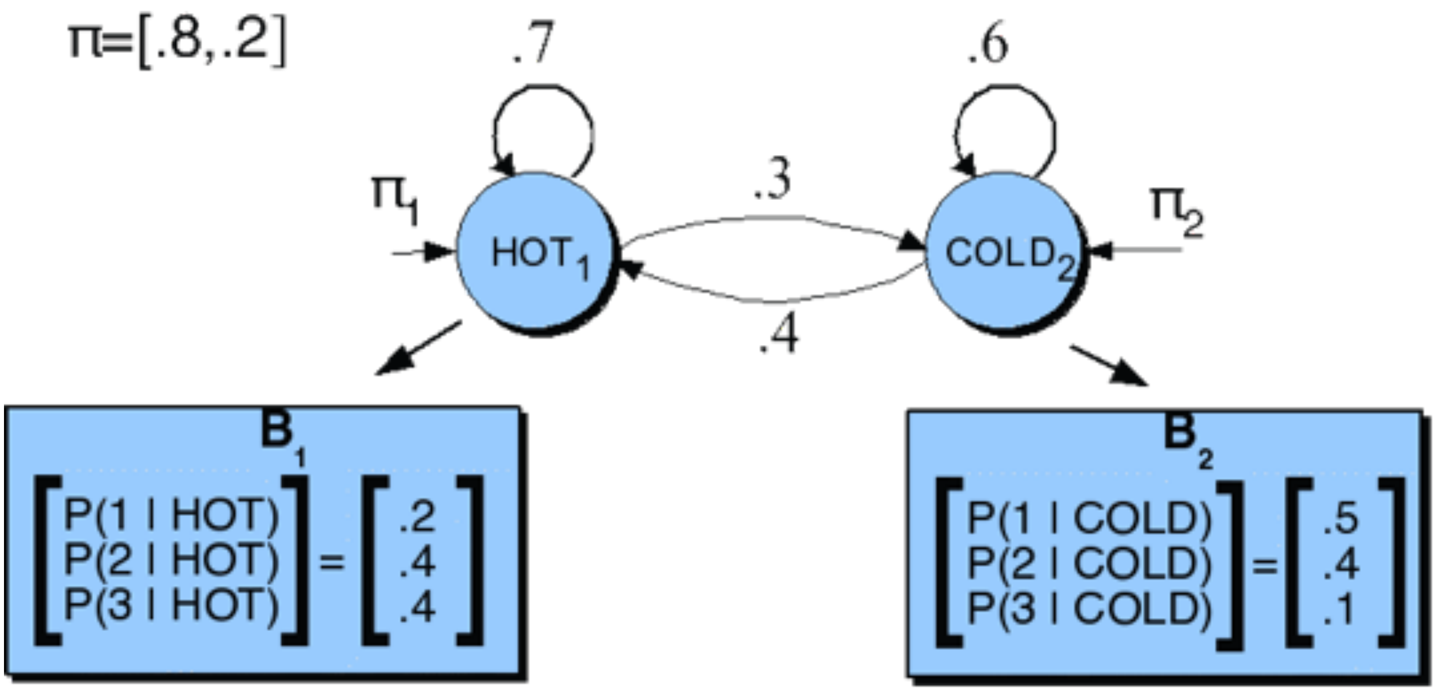
Weather and Ice Cream

- You are a climatologist in the year 2799 studying global warming
- You can't find any records of the weather in Claremont for summer of 2015
- But you find Maria Klawe's diary
- Which lists how many ice-creams Maria ate every day that summer
- Your job: Determine (hidden) sequence of weather states that `led to' Maria's (observed) ice cream behavior

Weather/Ice Cream HMM

- Hidden States: {Hot,Cold}
- Transition probabilities (A Matrix) between H and C
- Observations: {1,2,3} # of ice creams eaten per day
- Goal: Learn observation likelihoods between observations and weather states (Output Matrix B) by training HMM on aligned input streams from a training corpus
- Result: trained HMM for weather prediction given ice cream information alone

Ice Cream/Weather HMM



What can HMMs Do?

- ***Likelihood***: Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O, \lambda)$: Given # ice creams, what is the weather?
- ***Decoding***: Given an observation sequence O and an HMM $\lambda = (A,B)$, discover the best hidden state sequence Q: Given seq of ice creams, what was the most likely weather on those days?
- ***Learning***: Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B

Decoding: The Viterbi Algorithm

- Decoding: Given an observation sequence O and an HMM $\lambda = (A,B)$, discover the ***best*** hidden state sequence of weather states in Q
  - E.g., Given the observations 3 – 1 – 1 and an HMM, what is the ***best*** (most probable) hidden weather sequence of {H,C}
- Viterbi algorithm
  - Dynamic programming algorithm
  - Uses a dynamic programming trellis to store probabilities that the HMM is in state j after seeing the first t observations, for all states j

– Value in each cell computed by taking MAX over all paths leading to this cell – i.e. best path

– Extension of a path from state i at time t-1 is computed by multiplying:

$$v_t(j) = \max_{1 \leq i \leq N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

– Most probable path is the max over all possible previous state sequences

HMM Training: The Forward-Backward (Baum-Welch) Algorithm

- ***Learning***: Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A (transition) and B (emission)
- Input: unlabeled seq of observations O and vocabulary of possible hidden states Q
  - E.g. for ice-cream weather:
    - Observations = {1,3,2,1,3,3,…}
    - Hidden states = {H,C,C,C,H,C,….}

- Intuitions
  - Iteratively re-estimate counts, starting from an initialization for A and B probabilities, e.g. all equi-probable
  - Estimate new probabilities by computing forward probability for an observation, dividing prob. mass among all paths contributing to it, and computing the backward probability from the same state
- Details: see e.g. Manning & Schütze

What is Syntax?

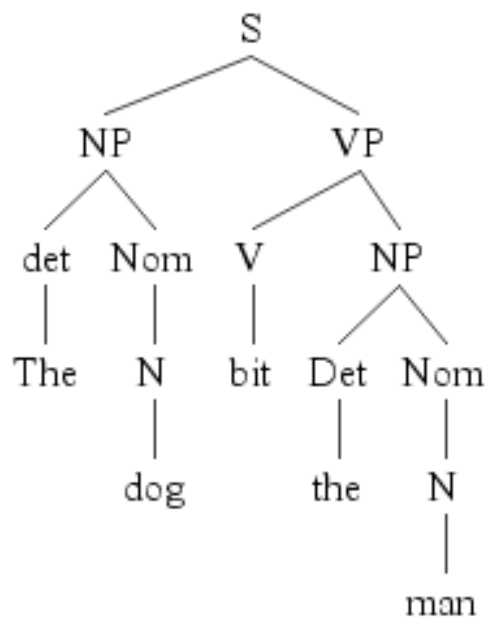
- The study of how the parts of an utterance are arranged in relation to one another
- Questions in syntax:
  - Do all languages behave the same way?
  - Can the structure of yet un-analyzed languages be predicted?
  - How is syntax learned by children, with little negative evidence?

Structural Descriptions

- A structure that shows word order, syntactic constituency, and labels for the constituents.
- Includes trees, bracketed structures

Structural Descriptions

- Tree



- Bracketed Structure

  [[The [dog]] [bit [the [man]]]]

- Labeled bracketed structure

  [S [NP [det The] [Nom [N dog]]] [VP [V bit] [NP [Det the] [Nom [N man]]]]]

Tree Structures

- Ordered directed trees with nodes, labels, arcs
  - Preterminal Node: Node with a single leaf as its descendant
    - What are preterminal nodes in NL grammar?
    - Part of Speech Tags
  - Arc: Shows constituency relation, but untyped
  - Label: Symbol giving the category of a node

What is Syntax? (part 2)

- Set of rules by which well formed utterances are formed.
- Formalize the notion of syntax: formal language theory

Formal Language Theory

- Natural language is rule-governed, not random
  - Like compilers, machine languages
  - Can construct a grammar to parse it
- Formal language theory
  - Conceptual framework for studying natural language.

Generalized Formal Grammar

- $G = \langle N,\Sigma,P,S \rangle$ where:
  - N is a set of non-terminal symbols, typically S,A,B,...
  - Σ is a set of terminals, typically x, y, z, . . .
  - P is a set of production rules
  - S is the starting or goal variable from N, i.e., $S \in N$

Sample Grammar

S → NP VP
NP → Det Noun
NP → ProperNoun
VP → Verb
VP → Verb NP
Det → the | a | that
Noun → lamp | pig | dirt
ProperNoun → Washington | Sam
Verb → understands | chases

Washington understands Sam

Sam chases that pig

*understands Sam

Context Free Grammar

$G = \langle N, \Sigma, P, S \rangle$ where:

- N is a set of non-terminal symbols, typically S,A,B,...
- S is the starting or goal symbol from N, i.e., $S \in N$
- Σ is a set of terminal symbols, typically x, y, z, . . . disjoint from N
- P is a set of production rules of the form $A \rightarrow \beta$, where:
  - A is a non-terminal $A \in N$
  - β is a string of symbols from $(\Sigma \cup N)$

CFGs for Natural Language

- A nonterminal symbol labels a syntactic part (constituent):
  NP, *VP*, *PP*, (*Noun*, *Verb*, *Det*)
- A starting symbol indicates which symbol has to come first; it labels the largest constituent or biggest part:
  S, *ROOT*, or *TOP*
- A terminal symbol labels the smallest part, the actual strings of the language:
  man, *they*, *swim*

CFGs for Natural Language

- Production rule (re-write rule): one symbol is rewritten ($\rightarrow$) as one or more others: NP $\rightarrow$ Det Noun
- A production rule captures the notion of syntactic constituency.
- 'LHS' is used to indicate the left-hand side of the $\rightarrow$, and likewise for 'RHS'.

Is this a valid CFG grammar?

NP $\rightarrow$ Det Noun
Nom $\rightarrow$ (Adj) Noun
VP $\rightarrow$ VB NP
Det $\rightarrow$ the | a

Noun $\rightarrow$ rabbit | carrots
Adj $\rightarrow$ fresh | crispy
VB $\rightarrow$ ate | likes

the rabbit likes crispy carrots.

Treebanks

- Linguistic corpora annotated for syntactic structure.
- Imply grammars of the languages they contain
- Examples:
  - The Penn Treebank (English)
  - Penn Chinese Treebank Project
  - The Tübingen Treebank of Written German
  - Arabic Treebank
  - Korean Treebank

Some Phrases in PTB

- NP: Noun phrase
- VP: Verb phrase
- PP: Prepositional phrase
- ADJP: Adjective phrase
- ADVP: Adverb phrase
- CONJP: Multi-word conjunctions ("not only")
- QP: quantifier phrase (inside NPs)
- 21 Total

Clauses in PTB

- S: declaratives, passives, imperatives, questions with declarative order, (embedded) infinitive clauses, gerund classes
- SINV: Inverted clauses
- SBAR: Relative and subordinate clauses
- SBARQ : Wh-questions
- SQ: Y/N-questions, inside SBARQ
- S-CLF : It-cleft clauses
- FRAG: Stand-alone clauses, phrases without a predicate argument structure

Rules in Treebanks

- Lots of them! 17,000 in PTB
  - Most very flat
  - Many tailored to single sentences
  - Number grows linearly with corpus
- Largest number: S, NP, VP

Two Goal of Parsing

Analyze input strings to assign proper structures

- For input A, grammar G:
  - Assign zero or more parse tree(s) T:
    - Cover all and only the elements of A
    - Root of T is S (the start symbol of G)
  - Do not necessarily pick one (or correct) analysis

Two Goal of Parsing

Recognition

- Subtask of parsing
- For input A, grammar G
  - Is A in the language defined by G?

Questions for Parsing

- Is this sentence in the language?
  - FSAs accept the regular languages defined by automaton
  - Parsers accept language defined by CFG
- What is the syntactic structure of this sentence?
  - Syntactic parse provides framework for semantic analysis
    - What is the subject?
    - Useful for e.g. question answering

Parsing as Search

- Search through possible parse trees
- Want one (or more) that derive input
- Formally, search problems are defined by:
  - Start state S,
  - Goal state G,
  - Successor Function:
    Transitions between states,
  - Path cost function

One Model of Parsing as Search

- Start State:
  - Start Symbol from grammar
- Goal test:
  - Does parse tree cover all and only input?
- Successor function:
  - Expand a non-terminal using production in grammar where non-terminal is LHS of grammar
- Path cost:
  - We'll ignore here

One Model of Parsing as Search

- Node: Partial solution to search problem:
  - Partial parse
- Search start node: Initial State
  - Input string
  - Start symbol of CFG
- Goal node:
  - Full parse tree: covering all and only input, rooted at S

Parse Search Strategies

- Two constraints:
  - Must start with the start symbol
  - Must cover exactly the input string
- Correspond to main parsing search strategies
  - Top-down search (Goal-directed search)
  - Bottom-up search (Data-driven search)

Parse Search Strategies

	Breadth-First	Depth-First
Top-Down		
Bottom-Up		