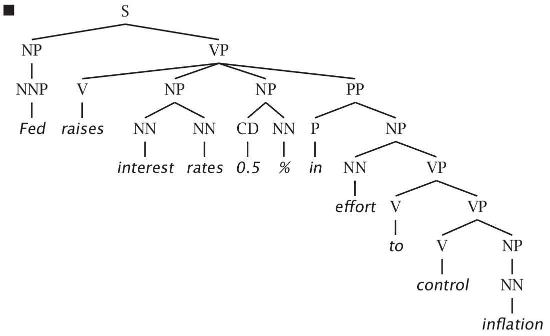
Part of Speech Tagging

FSNLP, chapters 9 and 10

Christopher Manning and Hinrich Schütze

© 1999–2002

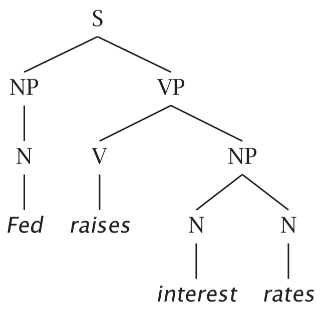
The problem of POS ambiguity

- Structures for: *Fed raises interest rates 0.5% in effort to control inflation* (*NYT* headline 17 May 2000)
- 

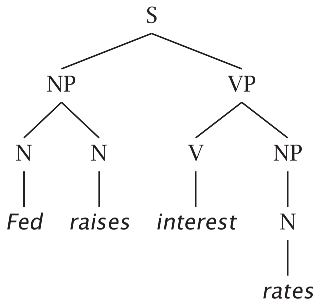
Part-of-speech ambiguities

		VB					
	VBZ	VBP	VBZ				
NNP	NNS	NN	NNS	CD	NN		
Fed	*raises*	*interest*	*rates*	*0.5*	*%*	*in*	*effort*
						to	*control*
							inflation

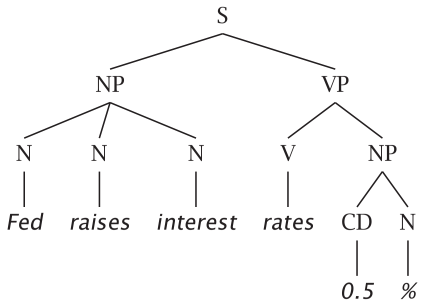
The bad effects of V/N ambiguities (1)



The bad effects of V/N ambiguities (2)



The bad effects of V/N ambiguities (3)



The task of part of speech tagging

- A lightweight (usually linear time) processing task, which can usefully empower other applications
  - Knowing how to pronounce a word: *récord* [noun] vs. *recórd* [verb]
  - Matching small phrasal chunks or particular patterns of word classes for tasks such as information extraction or collocation extraction. Ee.g., just matching nouns, compound nouns, and adjective noun patterns:
    - {A|N} N
  - POS information can be used to lemmatize a word correctly (i.e., to remove inflections):
    - *saw* [n] → *saw*; *saw* [v] → *see*

The task of part of speech tagging

-
  - Can differentiate word senses that involve part of speech differences
  - POS can be used as backoff in various class-based models, when too little information is known about a particular word
  - A preprocessor for a parser (often better, but more expensive, to let the parser do the tagging as well)
  - Tagged text helps linguists find interesting syntactic constructions in texts

The linguistics of parts of speech and tag sets

- We're not going to substantively discuss parts of speech in class
  - Read section 3.1 to learn about parts of speech, particularly the kind of richer sets of distinctions commonly made by linguists and in NLP applications
  - Read section 4.3.2 for discussion of POS tag sets used in NLP.
    - There's a handy table explaining tag abbreviations on pp. 141–142

Part of speech tagging

Information sources:

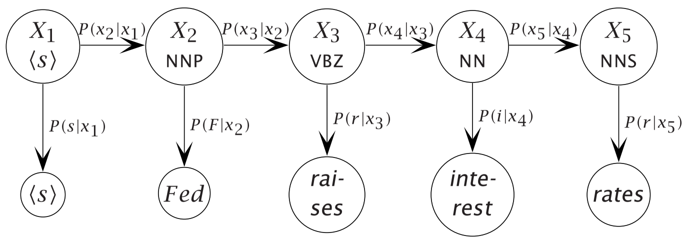
- Sequence of words: syntagmatic information
  - Surprisingly weak information source
  - Many words have various parts of speech – cf. the example above
- Frequency of use of words
  - Surprisingly effective: gets 90% performance by itself (for English)
    - This acts as a baseline for performance

(Hidden) Markov model tagger

- View sequence of tags as a Markov chain. Assumptions:
  - **Limited horizon.** $P(X_{i+1} = t^j | X_1, \ldots, X_i) = P(X_{i+1} = t^j | X_i)$
  - **Time invariant (stationary).** $P(X_{i+1} = t^j | X_i) = P(X_2 = t^j | X_1)$

  We assume that a word's tag only depends on the previous tag (limited horizon) and that this dependency does not change over time (time invariance)
- A state (part of speech) generates a word. We assume it depends only on the state

Hidden Markov Models – POS example



- Top row is unobserved states, interpreted as POS tags
- Bottom row is observed output observations
- We normally do supervised training, and then (Bayesian network style) inference to decide POS tags

Standard HMM formalism

- $\langle X, O, \Pi, A, B \rangle$
- X is hidden state sequence; O is observation sequence
- Π is probability of starting in some state
  (can be folded in: let $A' = [\Pi|A]$, i.e., $a_{0j} = \pi_j$)
- A is matrix of transition probabilities (top row CPTs)
- B is matrix of output probabilities (vertical CPTs)

HMM is also a probabilistic (nondeterministic) finite state automaton, with probabilistic outputs (from vertices, not arcs, in the simplest case)

Probabilistic inference in an HMM

Three fundamental questions:

- Compute the probability of a given observation sequence
- Given an observation sequence, compute the most likely hidden state sequence
- Given an observation sequence and set of possible models, which model most closely fits the data?

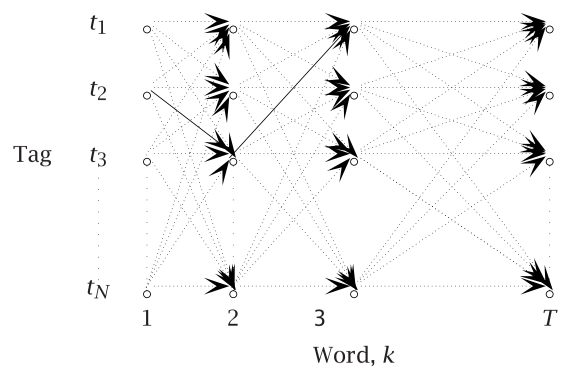
Probability of an observation sequence

- Given $O = (o_1, \ldots, o_T)$ and model $\mu = (A, B, \Pi)$
- $P(O|X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \cdots b_{x_T o_T}$
- $P(X|\mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \cdots a_{X_{T-1} X_T}$
- $P(O, X|\mu) = P(O|X, \mu) P(X|\mu)$
- $P(O|\mu) = \sum_X P(O|X, \mu) P(X|\mu)$ [Marginalization]
- $P(O|\mu) = \sum_{x_1 \cdots x_T} \prod_{t=1}^{T} a_{X_{t-1} X_t} b_{x_t o_t}$
- Difficult to compute like this!

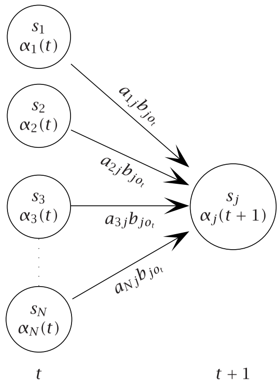
Dynamic Programming

- Efficient computation of this quantity: forward procedure
- Intuition: Probability of the first t observations is the same for all possible $t+1$ length state sequences.
- Define forward probability
  $\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, X_t = i|\mu)$
- $\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{j o_t}$
- Compute it recursively from beginning
- Similarly calculated backward probability $\beta_i(t)$ from end
- A version of the variable elimination algorithm for Bayes Net inference

Trellis algorithms



Closeup of the computation at one node



$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{j o_t}$$

Viterbi algorithm

- Used to efficiently find the state sequence that gives the highest probability to the observed outputs
- Another dynamic programming algorithm. Essentially the same except you do a max instead of a summation, and record the path taken:

$$\delta_{i+1}(t^j) = \max_{1\le k\le T}[\delta_i(t^k) \times P(t^j|t^k) \times P(w_{i+1}|t^j)]$$

$$\psi_{i+1}(t^j) = \arg\max_{1\le k\le T}[\delta_i(t^k) \times P(t^j|t^k) \times P(w_{i+1}|t^j)]$$

- This gives a tag sequence for POS tagging

(H)MM POS tagging

- Estimate a and b parameters, preferably by doing counts from tagged training data, or else by using lexicons, Baum-Welch, as best one can
  - i.e., in simplest case, train as visible Markov model, and only tag as hidden Markov model
- Use Viterbi algorithm to tag unseen text
- On Penn tag set, can get accuracy of up to 96.7% (Brants 2000)
- Many other methods have also been used for POS tagging, the best having similar accuracy

Tagging: Other topics

- Trigram taggers, variable memory models, etc.
- Smoothing
- Unknown words
  - Need to model via word features
- Building taggers without a hand-tagged corpus
- Other methods of tagging:
  - Transformation based learning
  - Decision trees
  - Maximum entropy models
  - . . .

Tagging: Other topics

- Other languages
- Accuracy (factors: training data, tag set, drift, unknown words)
- Linguistic foundations of tags: not always determinate
- Applications: IR, IE, QA, terminology extraction