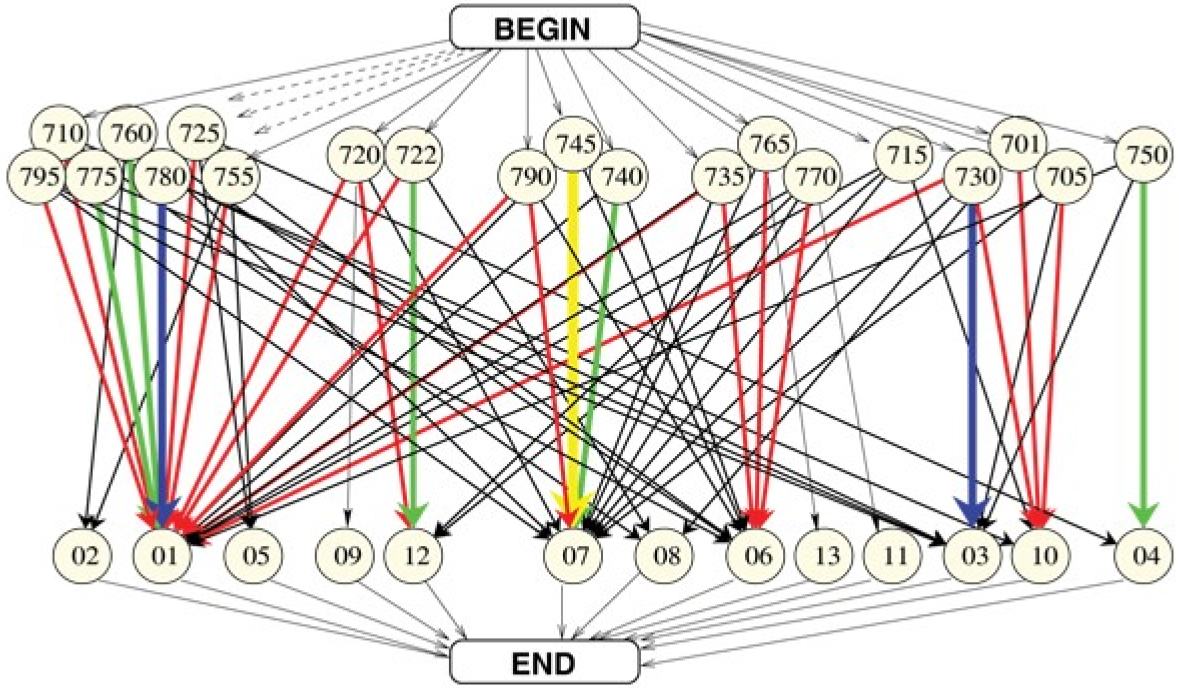
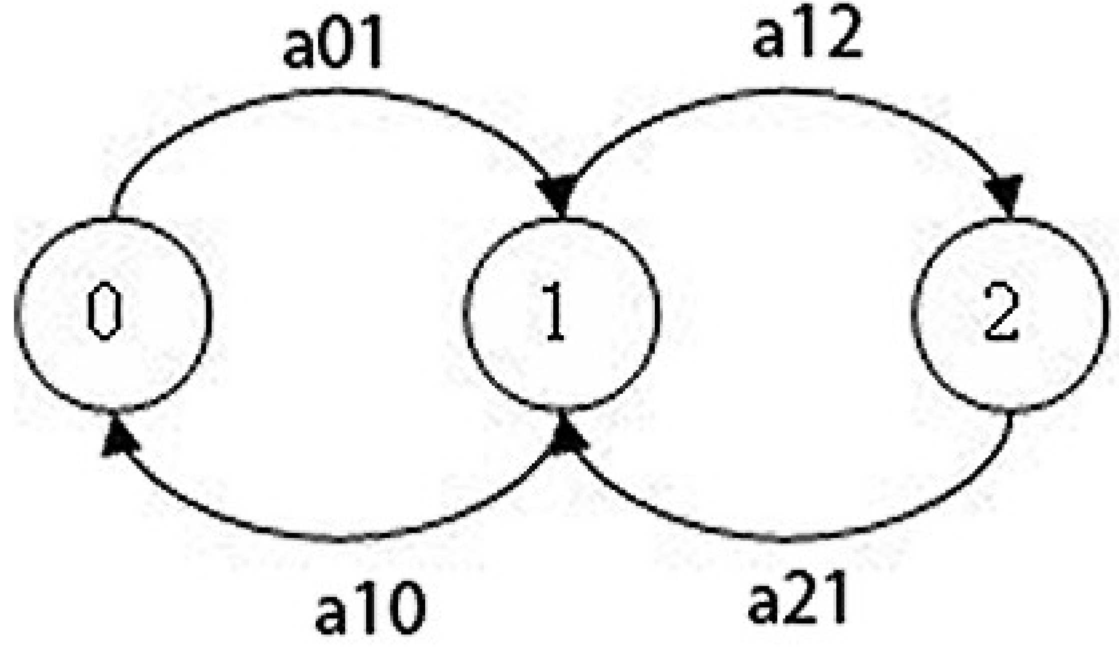
Hidden Markov Model



Markov Chain

- Markov chain is a discrete random process with the Markov property.
- The Markov property states that the probability distribution for the system at the next step (and in fact at all future steps) only depends on the current state of the system, and not additionally on the state of the system at previous steps.

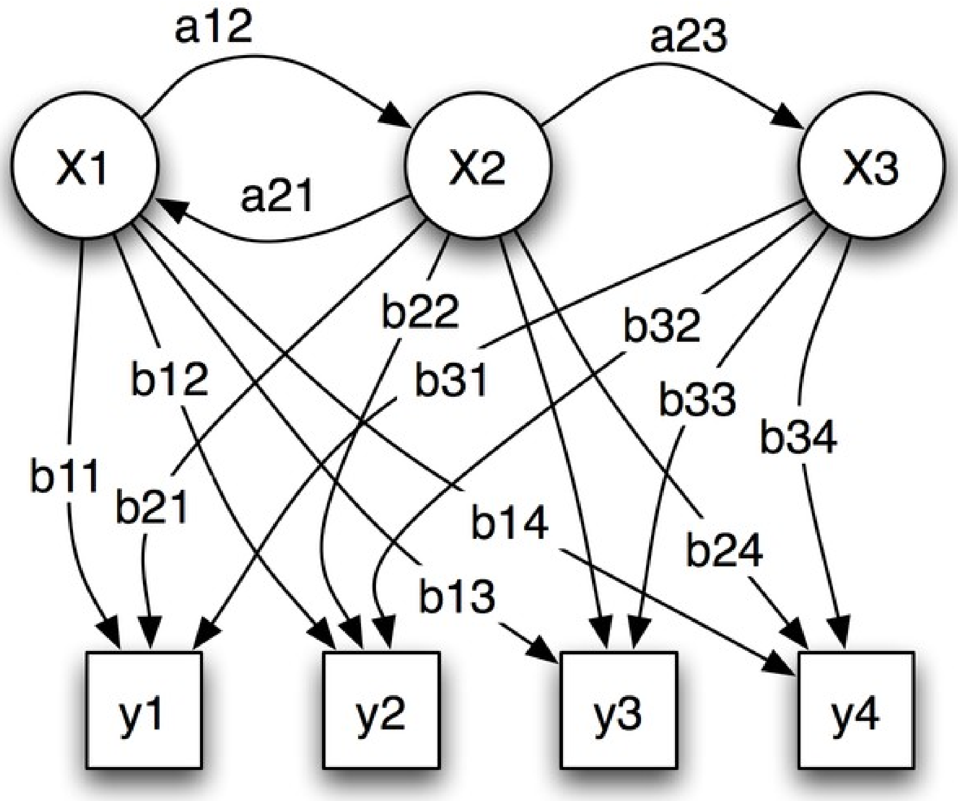
Markov Chain



Hidden Markov Model

- In a regular Markov model, the state is directly visible to the observer.
- In a hidden Markov model, the state is not directly visible, but output dependent on the state is visible.
- Each state has a probability distribution over the possible output tokens.

Hidden Markov Model



- x – states
- y – observations
- a – state transition probabilities
- b – output probabilities

Hidden Markov Model

More formally...

The probability to observe a sequence of length N:

$$Y = y(0), y(1), \dots , y(N-1)$$

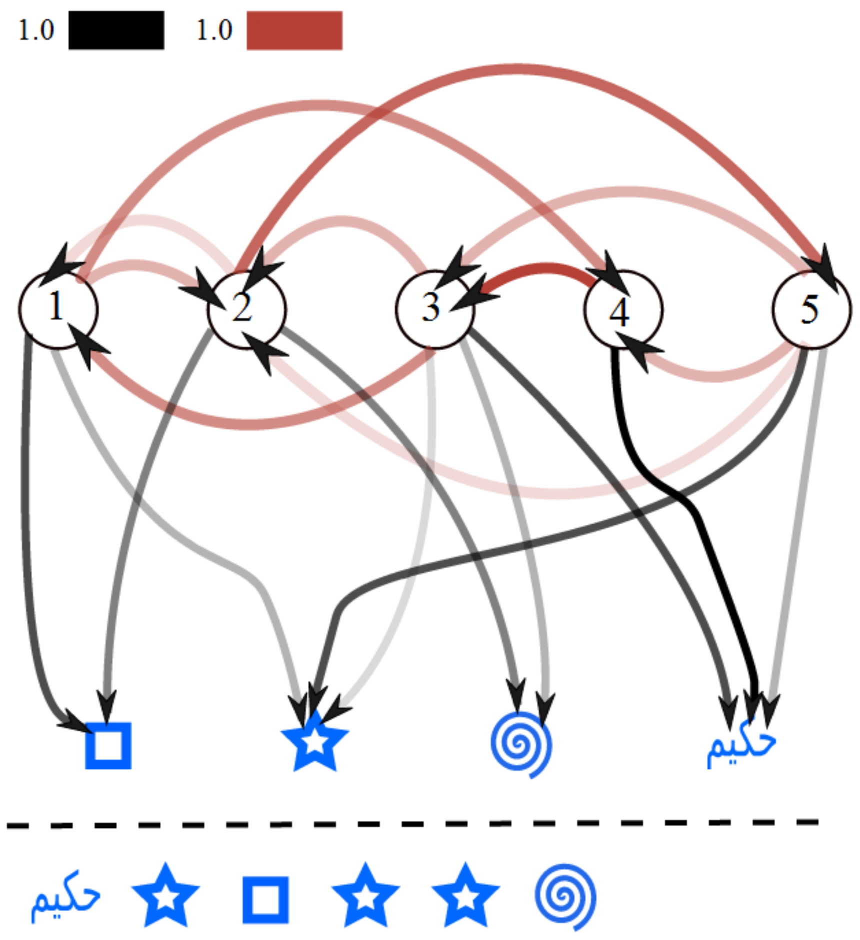
is

$$P(Y) = \sum_{X} P(Y|X)P(X)$$

where the sum runs over all possible hidden-node sequences

$$X = x(0), x(1), \dots , x(N-1)$$

Hidden Markov Model



The output can be produced by several sequences:

5 3 2 5 3 2

4 3 2 5 3 2

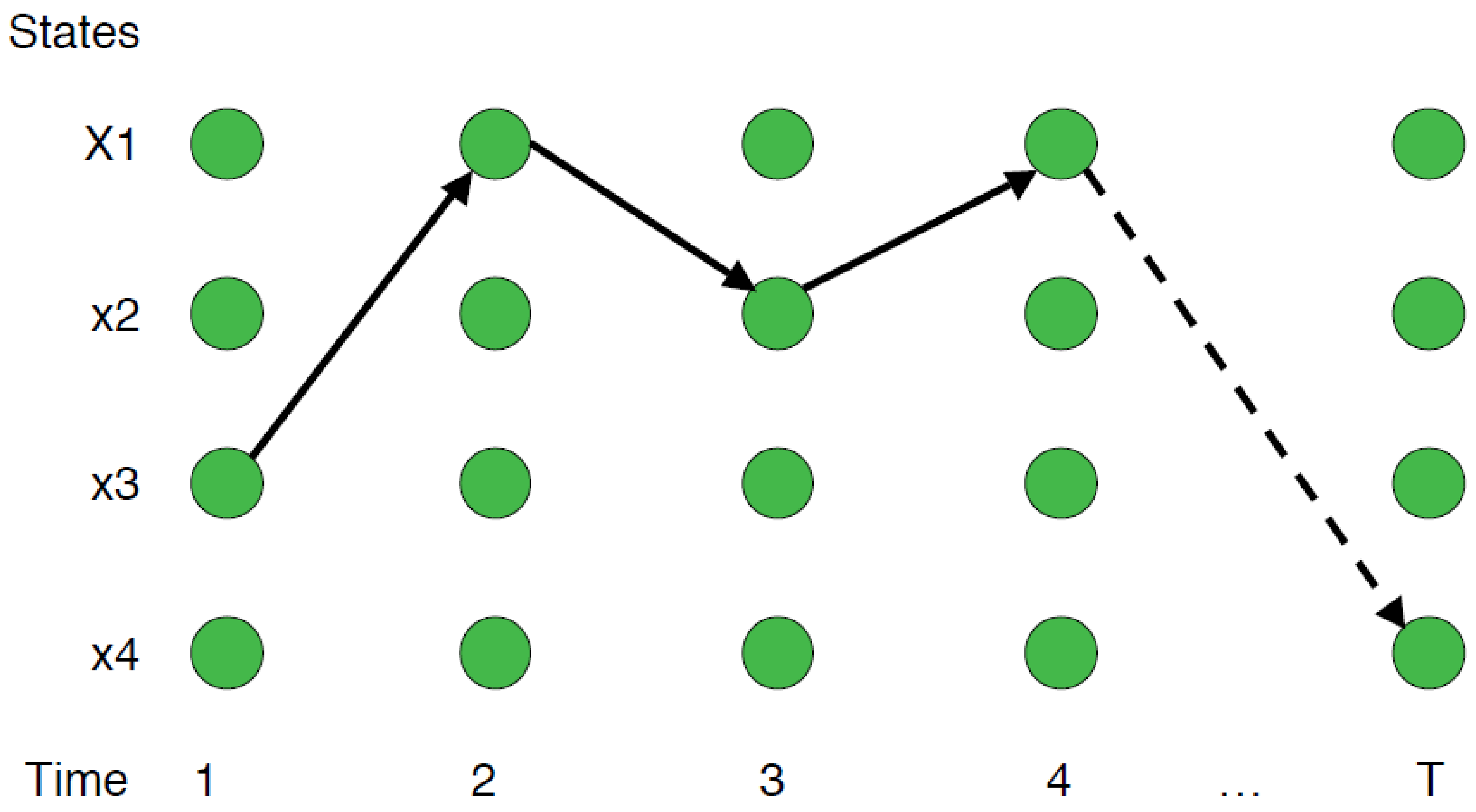
3 1 2 5 3 2

Hidden Markov Model

- Brute force calculation of P(Y) is insane for most real-life problems. The number of possible hidden node sequences scales exponentially with the length of the output sequence.
- Algorithms exist to help us depending on the nature of our problem.

Hidden Markov Model

The possibilities...



Hidden Markov Model
Questions for a HMM

- “Given the parameters of the model, compute a probability of a particular output sequence”
- “Given the parameters of the model and a particular output sequence, find a state sequence that is most likely to have generated that output sequence”
- “Given an output sequence (or a set), find the most likely set of state transition and output probabilities”

Hidden Markov Model

Applications:

- Speech recognition
- Handwriting recognition
- Machine translation
- Part-Of-Speech (POS) Tagging
- Gene prediction
- etc...

Viterbi Algorithm

- The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states (the Viterbi path) that results in a sequence of observed events.

Viterbi Algorithm

We need to maintain two dynamic programming tables:

- Probability of the best path:

$$\delta_j(t+1) = \max_{i=1..N} \delta_i(t) a[x_j|x_i]\, b[o_{t+1}|x_j]$$

- State transitions of the best path.
- Compute recursively from the beginning.

Viterbi Algorithm

- Memory efficient. We only need to store the best path.
- Fast, and therefore, practical.

Hidden Markov Model
Part-Of-Speech

- Represent a sequence of tags as a Markov Chain.
- Observations Y: sequence of words $y_1,\ldots,y_N$
- Goal: find the most probable sequence of tags:

  $x_1,\ldots,x_N$

HMM POS

Assumptions:

- Limited Horizon $P(x_{t+1}|x_1,...,x_n) = P(x_{n+1}|x_n)$

  (words are chosen independently from each other)
- Time Invariant (Stationary) $P(x_{n+1}|x_n) = P(x_2|x_1)$

  (dependency does not change over time)
- A state (POS) generates a word. We assume it depends only on one state.

  $P(y_n|x_1,...,x_N,y_1,...y_{n-1}) =P(y_n|x_n)$

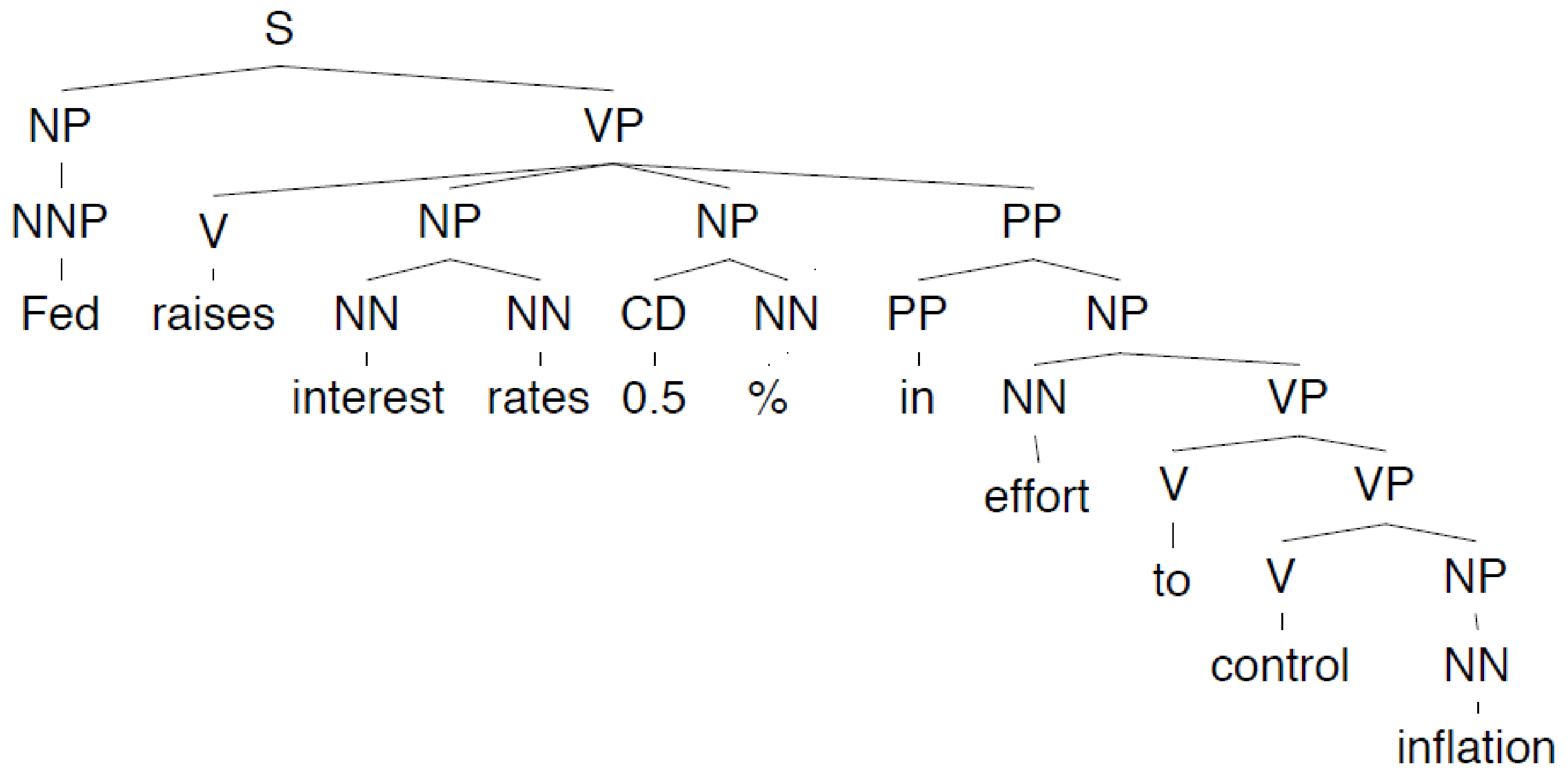
HMM POS

Let us consider the following sentence:

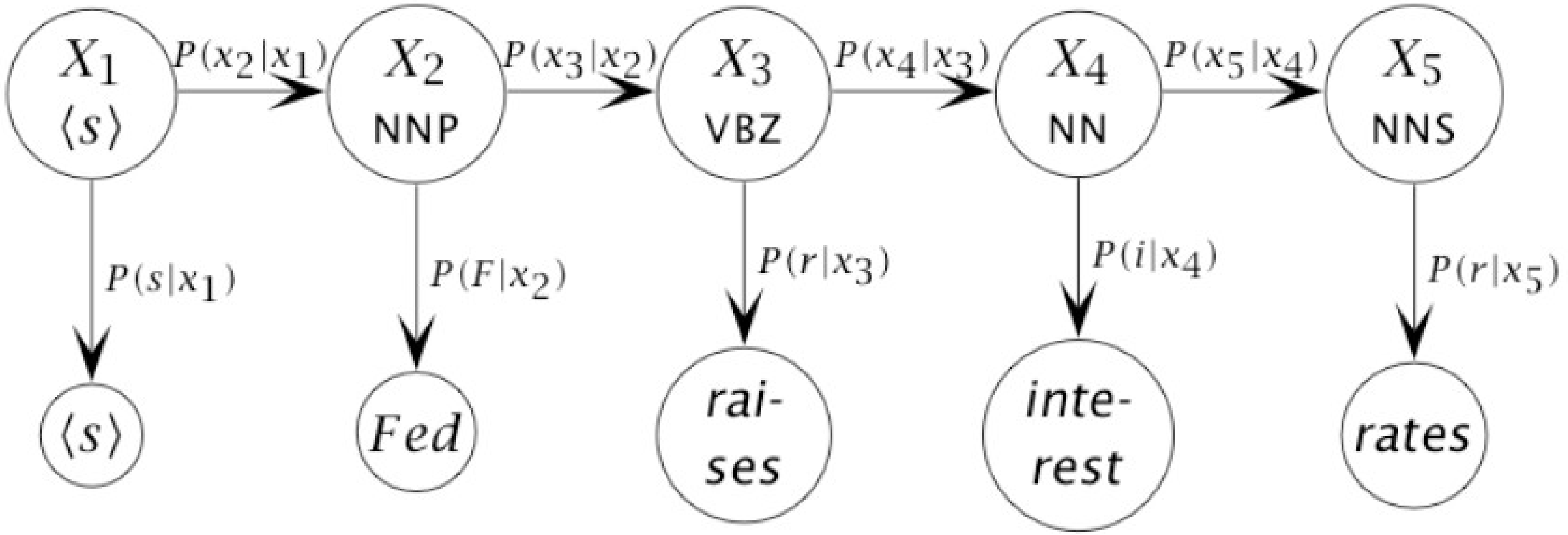
“Fed raises interest rates 0.5% in effort to control inflation.”

NY Times Headline17 May 2000

HMM POS


S
NP
VP
NNP
V
NP
NP
PP
Fed
raises
NN
NN
CD
NN
PP
NP
interest
rates
0.5
%
in
NN
VP
effort
V
VP
to
V
NP
control
NN
inflation

HMM POS



Top row (hidden states) are POS tags.

Bottom row (words) are output observations.

Laplace Smoothing

- Estimating probabilities works well when you have a lot of data. But lexical probabilities have very sparse distributions. Some kind of smoothing is required.
- The Brown corpus contains about 1,000,000 words. But it contains only 49,000 unique words and over 40,000 of those words occur less than 5 times!
- Laplace Smoothing: Add 1 to all frequency counts to pretend we've seen everything once, and then re-normalize the probability space.

Laplace Smoothing

- Let $N = \sum_{i}^{|V|} c(w_i)$ where V = vocabulary size
- Without smoothing: $P(w_x) = \frac{c(w_x)}{N}$
- We compute the vocabulary size for each term:

$$new\_c(w_x) = (c(w_x) + 1) * \frac{N}{N+V}$$

- New probability estimates:

$$new\_P(w_x) = \frac{new\_c(w_x)}{N}$$

Hidden Markov Model
Part-Of-Speech

Findings:

- English language tagging accuracy of 96%.
- Chinese language tagging accuracy of 94.78%.

References

- F. M. Hasan, N. UzZaman and M. Khan, “Comparison of different POS Tagging Techniques (N-Gram, HMM and Brill’s tagger) for Bangla”
- Z. Huang, V. Eidelman, M. Harper, “Improving A Simple Bigram HMM Part-of-Speech Tagger by Latent Annotation and Self-Training”
- S. M. Thede, M. P. Harper, “A Second-Order Hidden Markov Model for Part-of-Speech Tagging”
- A. W. Moore, “Hidden Markov Models”, Carnegie Mellon University
- Wikipedia, “Hidden Markov Model”, http://en.wikipedia.org/wiki/Hidden_Markov_model

Fin!

Questions?