CMPT-413: Computational Linguistics

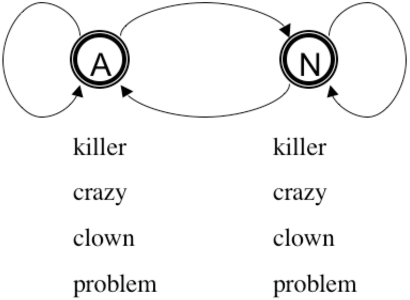
HMM3: Parsing with Hidden Markov Models

Anoop Sarkar
http://www.cs.sfu.ca/∼anoop

February 28, 2013

Hidden Markov Model

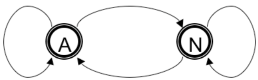
$$\text{Model } \theta = \begin{cases} \pi_i & \text{probability of starting at state } i \\ a_{i,j} & \text{probability of transition from state } i \text{ to state } j \\ b_i(o) & \text{probability of output } o \text{ at state } i \end{cases}$$



Hidden Markov Model Algorithms

- HMM as parser: compute the best sequence of states for a given observation sequence.
- HMM as language model: compute probability of given observation sequence.
- HMM as learner: given a corpus of observation sequences, learn its distribution, i.e. learn the parameters of the HMM from the corpus.
  - Learning from a set of observations with the sequence of states provided (states are not hidden) [Supervised Learning]
  - Learning from a set of observations without any state information. [Unsupervised Learning]

HMM as Parser



$\pi =$

A	0.25
N	0.75

$a =$

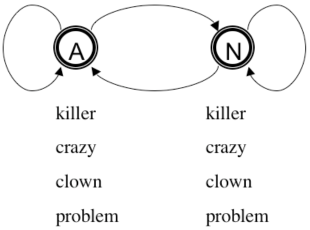
$a_{i,j}$	A	N
A	0.0	1.0
N	0.5	0.5

$b =$

$b_i(o)$	*clown*	*killer*	*problem*	*crazy*
A	0	0	0	1
N	0.4	0.3	0.3	0

The task: for a given observation sequence find the most likely state sequence.

HMM as Parser



- Find most likely sequence of states for *killer clown*
- Score every possible sequence of states: AA, AN, NN, NA
  - P(killer clown, AA) $= \pi_A \cdot b_A(killer) \cdot a_{A,A} \cdot b_A(clown) = 0.0$
  - P(killer clown, AN) $= \pi_A \cdot b_A(killer) \cdot a_{A,N} \cdot b_N(clown) = 0.0$
  - P(killer clown, NN) $= \pi_N \cdot b_N(killer) \cdot a_{N,N} \cdot b_N(clown) =$ $0.75 \cdot 0.3 \cdot 0.5 \cdot 0.4 = 0.045$
  - P(killer clown, NA) $= \pi_N \cdot b_N(killer) \cdot a_{N,A} \cdot b_A(clown) = 0.0$
- Pick the state sequence with highest probability (NN=0.045).

HMM as Parser

- As we have seen, for input of length 2, and a HMM with 2 states there are 2^2 possible state sequences.
- In general, if we have q states and input of length T there are q^T possible state sequences.
- Using our example HMM, for input *killer crazy clown problem* we will have 2^4 possible state sequences to score.
- Our naive algorithm takes exponential time to find the best state sequence for a given input.
- The **Viterbi algorithm** uses dynamic programming to provide the best state sequence with a time complexity of $q^2 \cdot T$