CSA4050 Advanced Techniques in NLP

Mike Rosner

November 2004

The Decoding Problem

- From a statistical point of view, both spelling correction and speech recognition are instances of the *decoding problem*: which underlying word might have produced an observation sequence O.
- Recall that the the Bayesian method of word identification requires the identification of
  - *Prior probability* of each word P(w)
  - *Likelihood* of O given the word P(O|w)).

  in order to maximise P(w) * P(O|w) for each candidate word.

Confusion Matrices: Limitations

- For the spelling correction problem, we computed likelihood by means of *confusion matrices* for for each type of error.
- Only useful for decoding mis-spellings that are the result of a single error (insertion, deletion, or substitution).
- They are of limited use for errors caused by *contextual factors*.
- This is most clearly seen when decoding speech.
- Pronunciation variation occurs for complex reasons e.g. context, lexical frequency, stress, intonation.

Pronunciation Variation

- Interpret the sequence of phones [ni] when it occurs after the word "I" at the beginning of a sentence.
- What words sound like [ni]?
- Investigation of the "Switchboard Corpus" reveals the following words are pronounced [ni]

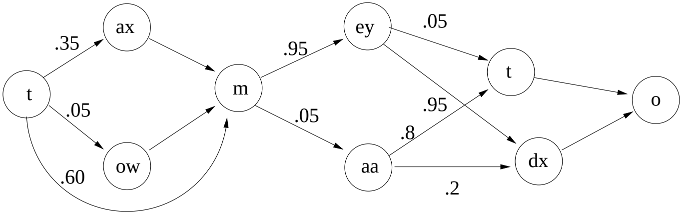
word	context
the	in *the* island
neat	*neat* little
need	*need* love
new	*New* York
knee	his *knee* hurts
to	talking *to* you

Decoding Writing versus Decoding Speech

- **Spelling Correction**
  - Input is fully determined (characters in the input string are known exactly).
  - Word boundaries are fully determined (mostly).
  - Given an input, candidate source words are typically *generated dynamically*.
- **Speech Perception**
  - Input is underdetermined (each element of input sequence associated with a probability).
  - Word boundaries are generally not known in advance.
  - Given an input, candidate source words are typically *precomputed and stored*.

Weighted Automata

- For purposes of efficiency a lexicon is often stored with the most likely kind of pronunciation variation pre-compiled.
- A common representation for such variations is the *weighted automaton*.
- The weighted automaton is a simple augmentation of a finite automaton in which each arc is associated with a probability.



Weighted Automata

- States correspond to observable segments of sound called *phones*.
- The probability of all arcs leaving a state must sum to 1.
- Each spanning path corresponds to a possible observation sequence for a word.

The Forward Algorithm

- The forward algorithm computes P(O|w) given
  - the observation sequence O
  - the weighted automaton for w and
- The forward algorithm is another dynamic programming algorithm and can be thought of as a slight generalisation of the minimum edit distance (MED) algorithm.
- Like the MED is uses a table to store intermediate values as it builds up the probability of the observation sequence.
- The Forward algorithm does not in itself solve the decoding problem

Forward Algorithm Informal Description

- Each cell of the table f[j,t] represents the probability of being in state j after seeing the first t observations
- The value of each cell f[j,t] is computed by summing over the probabilities of every path that could lead to this cell, by summing over the extensions of all the paths that lead to the current cell.
- The probability of an extension of a path from a state i at time t to a state j at time t+1 is computed by by multiplying together the following three factors:
  - The current path probability from the cell f[i,t].
  - The transition probability a[i,j] from state i to state j.
  - The observation likelihood b[j,t]that state j matches observation symbol O[t]. Here we assume that b[j,t] may take the value 1 or 0.

The Forward Algorithm

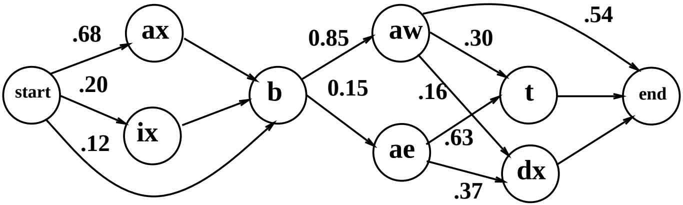
```
function Forward(O,w) returns forward-probability
 nstates = number of states in w
 nobs = length(O)
 new array f[nstates+2,nobs+2]

 f[0,0] = 1.0
 for t from 0 to nobs
  for i from 0 to nstates
    for each transition j from i in w
      f[j,t+1] = f[j,t+1] + f[s,t]*a[i,j]*b[j,O[t]]
return sum of probabilities in the final column of f
```

Result of Forward on [ni] for “need”

		#	n	iy	#
	end				.00056* .11 = .00062
	d				
need	iy			.00056 * 1.0 = .00056	
	n		.00056 * 1.0 = .00056		
	start	1.0			

Differences between MED and Forward

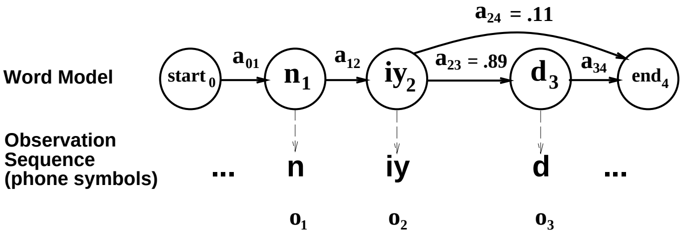
- Unlike the MED algorithm, we cannot calculate the value of a cell from the three cells around it. A state might be entered from any other, so the recurrence relation will be more complex.



- MED picks the *minimum cost* of the three possible routes to a given cell, whilst
- Forward computes the *sum* of the probabilities of all possible paths that could generate the observation sequence.

How Forward relates to Decoding

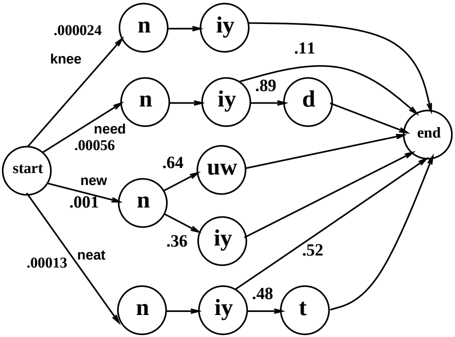
- Given an observation sequence O, Forward evaluates the probability of a word by calculating the *sum* of the probabilities of each spanning path through the automaton consistent with O.



- Thus one way to solve the decoding problem given O is to run the forward algorithm separately on each word and choose the word with the highest value.

A Combined Automaton

- A more realistic setup for a problem uses an automaton that combines word models for several words at once as shown below:



- The *Viterbi* algorithm can be used to choose the most probable word.

The Viterbi Algorithm

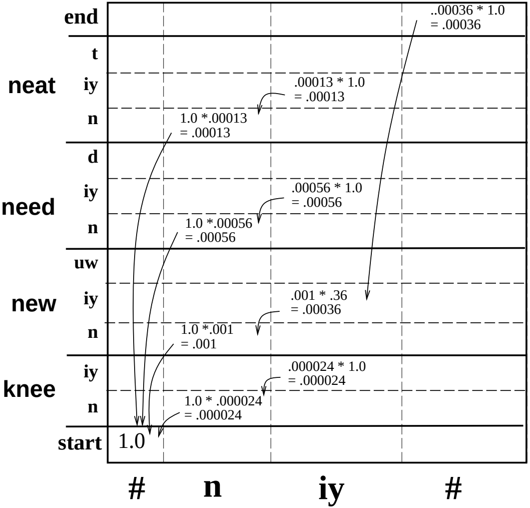
- The Viterbi algorithm is a variation of the Forward algorithm and is based on similar data structures.
- The two main differences are as follows:
  - At each step, Forward determines puts the *sum* of the probilities of all the previous paths into the current cell. Viterbi simply maintains the *maximum* value of the previous paths.
  - Whenever such a maximum value is determined, the path back to the previous state is maintained using a *back-pointer* array.

The Viterbi Algorithm

```
function Viterbi(O,W) returns best path through W
 nstates = number of states in W
 nobs = length(O)
 new array v[nstates+2,nobs+2]

 v[0,0] = 1.0
 for t from 0 to nobs
  for i from 0 to nstates
    for each transition s.t. a[i,j]>0 from i in W
      tmp = v[s,t]*a[i,j]*b[j,O[t]]
      if tmp > v[j,t+1]
      then { v[j,t+1] = tmp
             bp[j,t+1] = i }
```

- bp[i,t] holds a back pointer to the highest probability state at t-1.

Result of Viterbi



Summary

- Both Forward and Viterbi accept the same arguments: an observation sequence O and a weighted automaton W.
- Forward returns a probability (the probability of a being in state i having observed O).
- Viterbi returns a path: the most likely path having observed O.
- N.B. We have immensely simplified the problem by assuming that the symbols making up O are fully determined.
- In a real speech application the interpretation of the symbols themselves is probabilistic.