

HMM Algorithms: Trellis and Viterbi

PA154 Language Modeling (5.2)

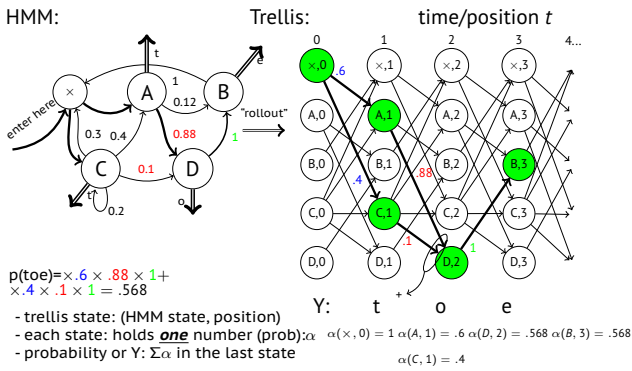
Pavel Rychlý

pary@fi.muni.cz

March 19, 2024

Source: Introduction to Natural Language Processing (600.465)
Jan Hajič, CS Dept., Johns Hopkins Univ.
www.cs.jhu.edu/~hajic

Trellis - Deterministic Output

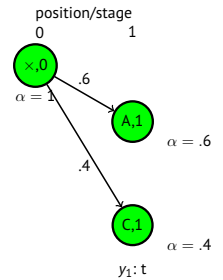


HMM: The Two Tasks

- HMM (the general case):
 - five-tuple (S, S_0, Y, P_s, P_Y) , where:
 - $S = \{s_1, s_2, \dots, s_T\}$ is the set of states, S_0 is the initial,
 - $Y = \{y_1, y_2, \dots, y_V\}$ is the output alphabet,
 - $P_s(s_j|s_i)$ is the set of prob. distributions of transitions,
 - $P_Y(y_k|s_i, s_j)$ is the set of output (emission) probability distributions.
 - Given an HMM & an output sequence $Y = \{y_1, y_2, \dots, y_k\}$
 - (Task 1) compute the probability of Y;
 - (Task 2) compute the most likely sequence of states which has generated Y.

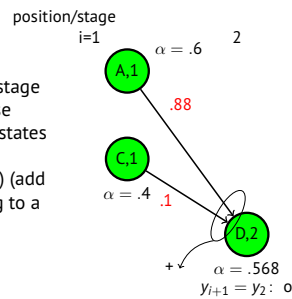
Creating the Trellis: The Start

- Start in the start state (\times),
 - its $\alpha(\times, 0) = 1$.
- Create the first stage:
 - get the first "output" symbol y_1
 - create the first stage (column)
 - but only those trellis states which generate y_1
 - set their $\alpha(state, 1)$ to the $P_s(state|\times) \times \underbrace{\alpha(\times, 0)}_1$
- ...and forget about the 0-th stage



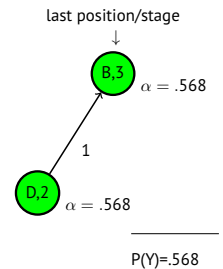
Trellis: The Next Step

- Suppose we are in stage i ,
- Creating the next stage:
 - create all trellis state in the next stage which generate y_{i+1} , but only those reachable from any of the stage- i states
 - set their $\alpha(state, i + 1)$ to: $P_s(state|prev.state) \times \alpha(prev.state, i)$ (add up all such numbers on arcs going to a common trellis state)
 - ...and forget about stage i



Trellis: The Last Step

- Continue until "output" exhausted
 - $|Y| = 3$: until stage 3
- Add together all the $\alpha(state_i|Y)$
- That's the $P(Y)$.
- Observation (pleasant):
 - memory usage max: $2|S|$
 - multiplications max: $|S|^2|Y|$

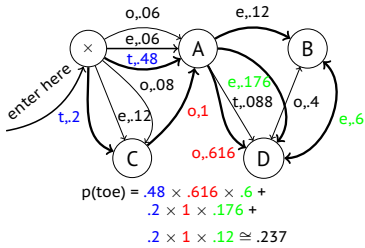


Trellis: The General Case (still, bigrams)

- Start as usual:
 - start state (x), set its $\alpha(x,0)$ to 1.

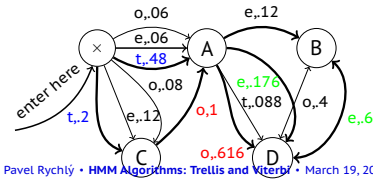
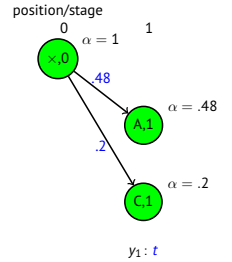


$\alpha = 1$



General Trellis: The Next Step

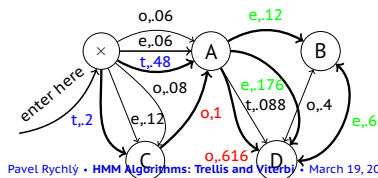
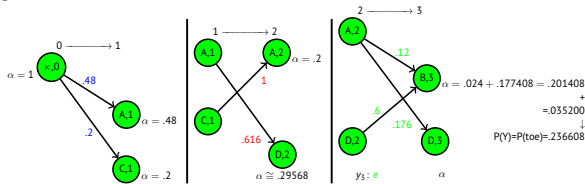
- We are in stage i :
 - Generate the next stage $i+1$ as before (except now arcs generate output, thus use only those arcs marked by the output symbol y_{i+1})
 - For each generated state compute $\alpha(\text{state}, i+1) = \sum_{\text{incoming arcs}} P_Y(y_{i+1} | \text{state}, \text{prev.state}) \times \alpha(\text{prev.state}, i)$



...and forget about stage i as usual

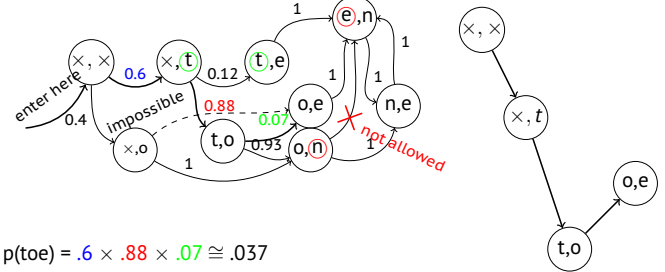
Trellis: The Complete Example

Stage:



The Case of Trigrams

- Like before, but:
 - states correspond to bigrams,
 - output function always emits the second output symbol of the pair (state) to which the arc goes:

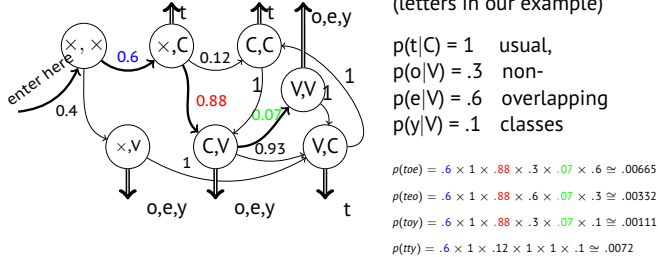


$p(\text{toe}) = .6 \times .88 \times .07 \cong .037$

Multiple paths not possible \rightarrow trellis not really needed

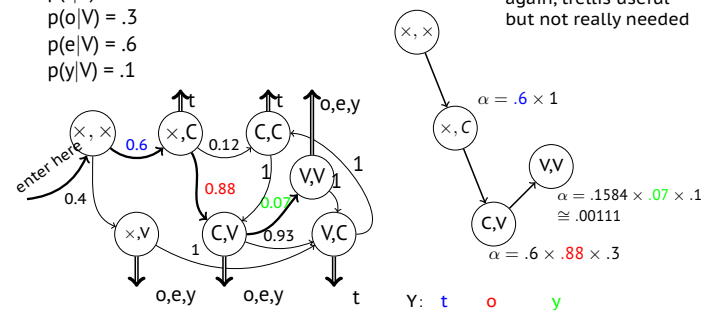
Trigrams with Classes

- More interesting:
 - n -gram class LM: $p(w_i | w_{i-2}, w_{i-1}) = p(w_i | c_i) p(c_i | c_{i-2}, c_{i-1})$
 - \rightarrow states are pairs of classes (c_{i-1}, c_i), and emit "words": (letters in our example)



Class Trigrams: the Trellis

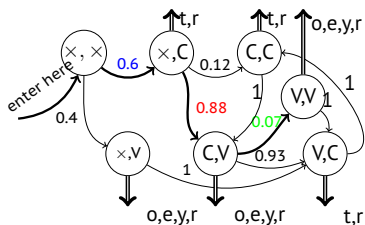
- Trellis generation ($Y = \text{"toy"}$):
 - $p(t|C) = 1$
 - $p(o|V) = .3$
 - $p(e|V) = .6$
 - $p(y|V) = .1$



Overlapping Classes

- Imagine that classes may overlap

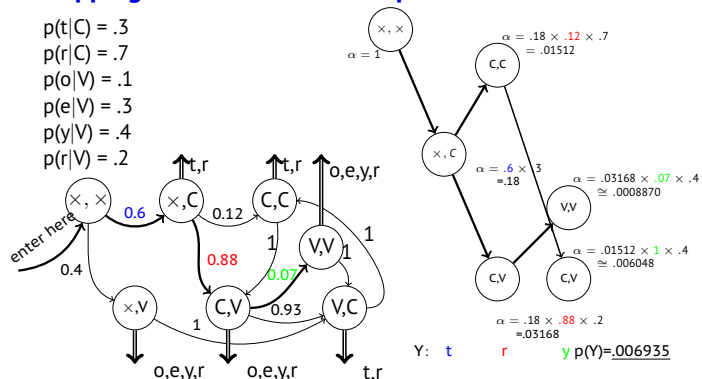
- e.g. 'r' is sometimes vowel sometimes consonant, belongs to V as well as C:



$$\begin{aligned}
 p(t|C) &= .3 \\
 p(r|C) &= .7 \\
 p(o|V) &= .1 \\
 p(e|V) &= .3 \\
 p(y|V) &= .4 \\
 p(r|V) &= .2
 \end{aligned}$$

$p(\text{try}) = ?$

Overlapping Classes: Trellis Example



Trellis: Remarks

- So far, we went left to right (computing α)
- Same result: going right to left (computing β)
 - supposed we know where to start (finite data)
- In fact, we might start in the middle going left and right
- Important for parameter estimation (Forward-Backward Algorithm alias Baum-Welch)
- Implementation issues:
 - scaling/normalizing probabilities, to avoid too small numbers & addition problems with many transitions

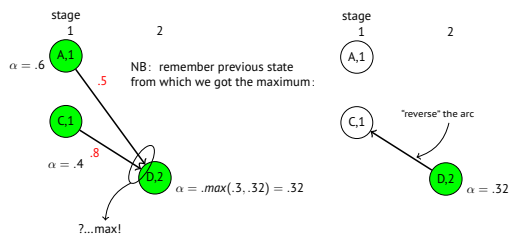
The Viterbi Algorithm

- Solving the task of finding the most likely sequence of states which generated the observed data
- i.e., finding

$$\begin{aligned}
 S_{best} &= \operatorname{argmax}_S P(S|Y) \\
 \text{which is equal to } (Y \text{ is constant and thus } P(Y) \text{ is fixed):} \\
 S_{best} &= \operatorname{argmax}_S P(S, Y) = \\
 &= \operatorname{argmax}_S P(s_0, s_1, s_2, \dots, s_k, y_1, y_2, \dots, y_k) = \\
 &= \operatorname{argmax}_S \prod_{i=1..k} P(y_i | s_i, s_{i-1}) P(s_i | s_{i-1})
 \end{aligned}$$

The Crucial Observation

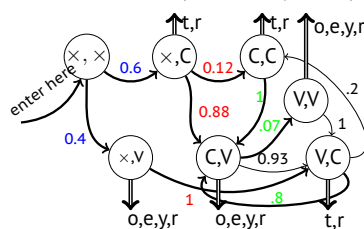
- Imagine the trellis build as before (but do not compute the α s yet; assume they are o.k.); stage i :



this is certainly the "backwards" maximum to (D.2)...but it cannot change even whenever we go forward (M. Property: Limited History)

Viterbi Example

- 'r' classification (C or V?, sequence?):

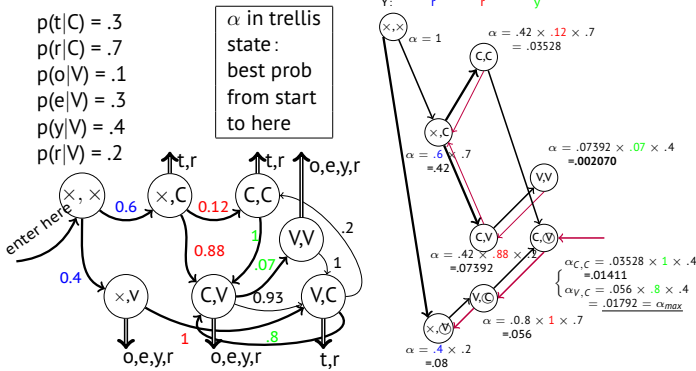


$$\begin{aligned}
 p(t|C) &= .3 \\
 p(r|C) &= .7 \\
 p(o|V) &= .1 \\
 p(e|V) &= .3 \\
 p(y|V) &= .4 \\
 p(r|V) &= .2
 \end{aligned}$$

$\operatorname{argmax}_{XYZ} p(\text{try}|XYZ) = ?$

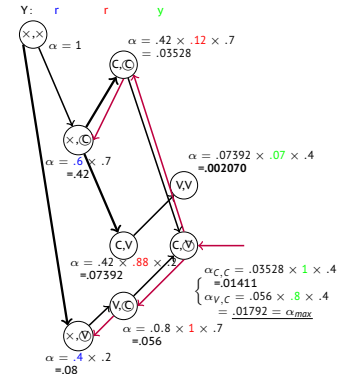
Possible state seq.: $(x, V)(V, C)(C, V)[VCV], (x, C)(C, C)(C, V)[CCV], (x, C)(C, V)(V, V)[CVV]$

Viterbi Computation



n-best State Sequences

- Keep track of n best "back pointers":
- Ex.: $n = 2$: Two "winners":
 - VCV (best)
 - CCV (2nd best)

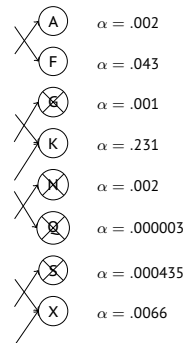


Tracking Back the n-best paths

- Backtracking-style algorithm:
 - Start at the end, in the best of the n states (s_{best})
 - Put the other $n-1$ best nodes/back pointer pairs on stack, except those leading from s_{best} to the same best-back state.
- Follow the back "beam" towards the start of the data, spitting out nodes on the way (backwards of course) using always only the best back pointer.
- At every beam split, push the diverging node/back pointer pairs onto the stack (node/beam width is sufficient!).
- When you reach the start of data, close the path, and pop the topmost node/back pointer(width) pair from the stack.
- Repeat until the stack is empty; expand the result tree if necessary.

Pruning

- Sometimes, too many trellis states in a stage:



- criteria: (a) $\alpha < \text{threshold}$
 (b) $\sum \pi < \text{threshold}$
 (c) # of states $>$ threshold (get rid of smallest α)