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

HMM: The Two Tasks

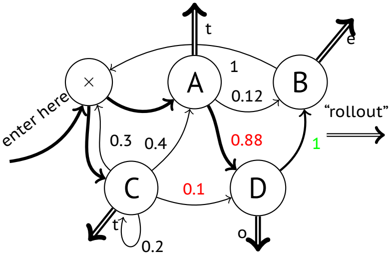
- HMM (the general case):
  - five-tuple (S, S_0, Y, P_s, P_Y), where:
    - S = $\{s_1, s_2, \ldots, s_T\}$ is the set of states, S_0 is the initial,
    - Y = $\{y_1, y_2, \ldots, y_v\}$ is the output alphabet,
    - $P_s(s_j|s_i)$ is the set of prob. distributions of transitions,
    - $P_Y(y_k|s_i, s_j)$ is the set of output (emission) probability distributions.
- Given an HMM & an output sequence $Y = \{y_1, y_2, \ldots, y_k\}$

  (Task 1) compute the probability of Y;

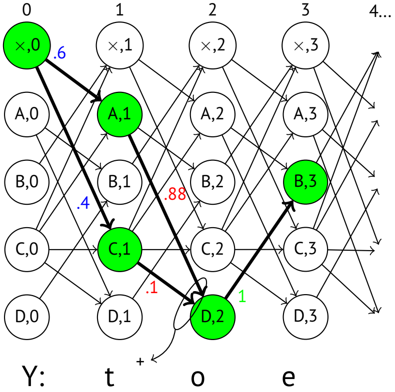
  (Task 2) compute the most likely sequence of states which has generated Y.

Trellis - Deterministic Output

HMM:



Trellis: time/position t



p(toe)=$\times .6 \times .88 \times 1 +$
$\times .4 \times .1 \times 1 = .568$

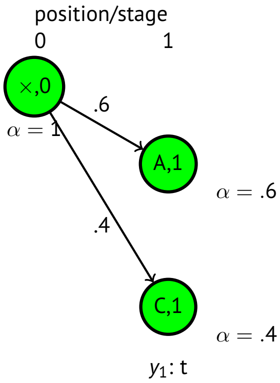
- trellis state: (HMM state, position)
- each state: holds ***one*** number (prob): α
- probability or Y: $\Sigma\alpha$ in the last state

$\alpha(\times, 0) = 1$ $\alpha(A, 1) = .6$ $\alpha(D, 2) = .568$ $\alpha(B, 3) = .568$

$\alpha(C, 1) = .4$

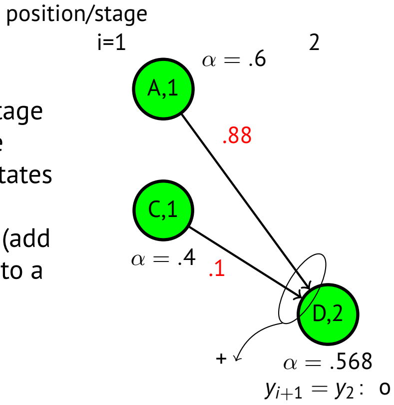
Creating the Trellis: The Start

- Start in the start state ($\times$),
  - its $\alpha(\times,0)$ to 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 $P_s(state|\times)\ \underbrace{\alpha(\times,0)}_{1}$
- ...and forget about the 0-th stage



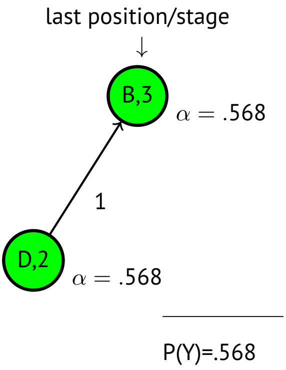
Trellis: The Next Step

- 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

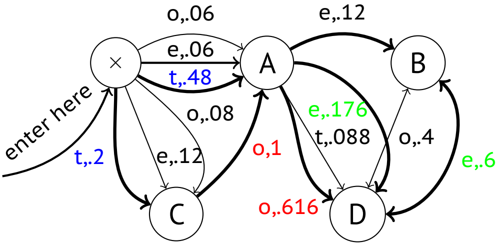
- Continue until "output" exhausted
  - $|Y| = 3$: until stage 3
- Add together all the $\alpha(state, |Y|)$
- That's the P(Y).
- Observation (pleasant):
  - memory usage max: $2|S|$
  - multiplications max: $|S|^2|Y|$



Trellis: The General Case (still, bigrams)

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

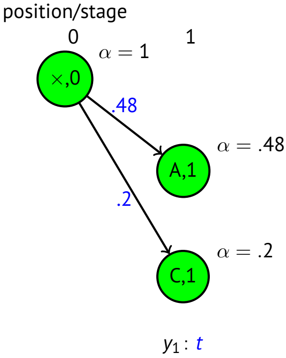
$\alpha = 1$

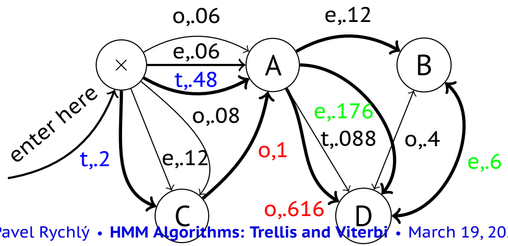


p(toe) = $.48 \times .616 \times .6 +$
$.2 \times 1 \times .176 +$
$.2 \times 1 \times .12 \cong .237$

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 $\alpha(state,i+1) =$
    $= \sum_{incoming\ arcs} P_Y(y_{i+1}|state,prev.state) \times \alpha(prev.state,i)$

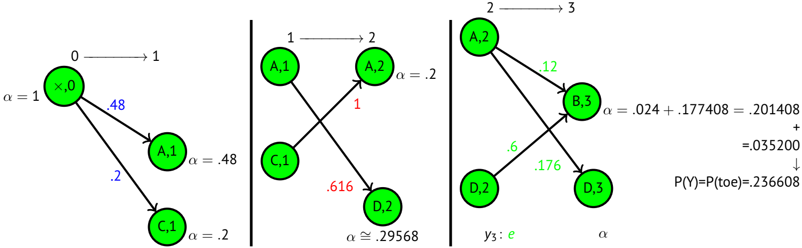


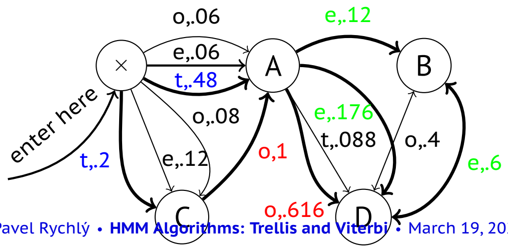


...and forget about stage *i* as usual

Trellis: The Complete Example

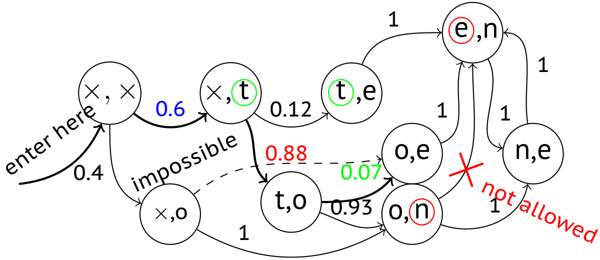
Stage:

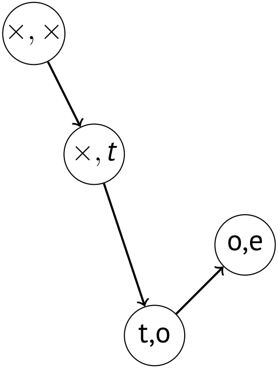




The Case of Trigrams

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





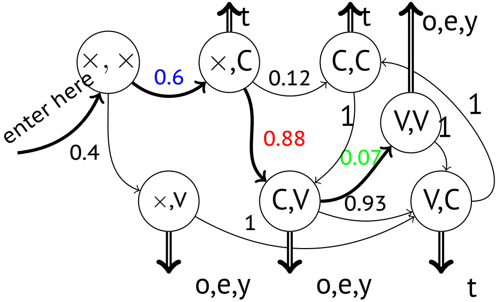
p(toe) = $.6 \times .88 \times .07 \cong .037$

Multiple paths not possible → 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)



p(t|C) = 1 usual,
p(o|V) = .3 non-
p(e|V) = .6 overlapping
p(y|V) = .1 classes

$p(toe) = .6 \times 1 \times .88 \times .3 \times .07 \times .6 \cong .00665$

$p(teo) = .6 \times 1 \times .88 \times .6 \times .07 \times .3 \cong .00332$

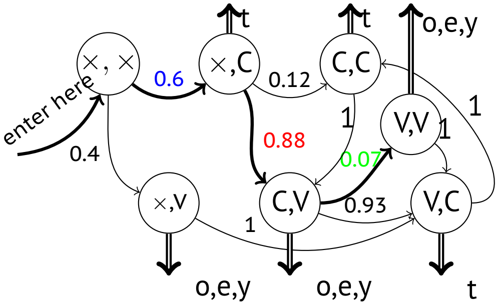
$p(toy) = .6 \times 1 \times .88 \times .3 \times .07 \times .1 \cong .00111$

$p(tty) = .6 \times 1 \times .12 \times 1 \times 1 \times .1 \cong .0072$

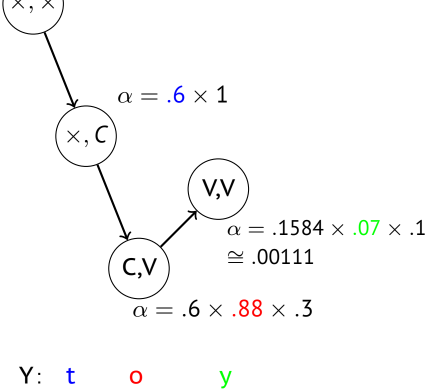
Class Trigrams: the Trellis

- Trellis generation (Y = “toy”):

  p(t|C) = 1
  p(o|V) = .3
  p(e|V) = .6
  p(y|V) = .1

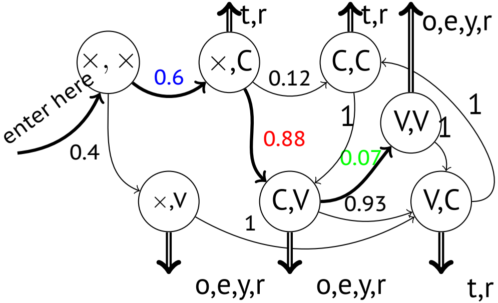


again, trellis useful
but not really needed



Overlapping Classes

- Imagine that classes may overlap
  - e.g. ‘r’ is sometimes vowel sometimes consonant, belongs to V as well as C:

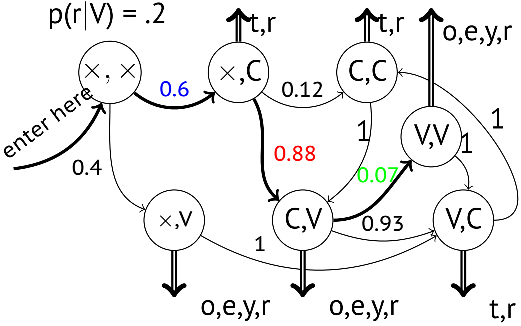


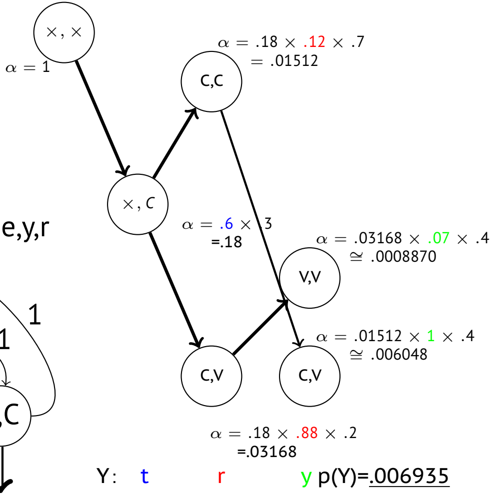
$p(t|C) = .3$
$p(r|C) = .7$
$p(o|V) = .1$
$p(e|V) = .3$
$p(y|V) = .4$
$p(r|V) = .2$

$p(try) = ?$

Overlapping Classes: Trellis Example

p(t|C) = .3
p(r|C) = .7
p(o|V) = .1
p(e|V) = .3
p(y|V) = .4
p(r|V) = .2





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 Algortihm 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

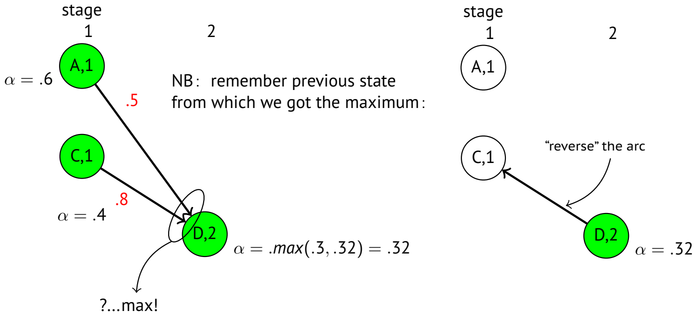
$$S_{best} = argmax_S \mathrm{P}(\mathrm{S}|\mathrm{Y})$$

which is equal to (Y is constant and thus P(Y) is fixed):

$$S_{best} = argmax_S \mathrm{P}(\mathrm{S,Y}) =$$
$$= argmax_S \mathrm{P}(s_0, s_1, s_2, \ldots, s_k, y_1, y_2, \ldots, y_k) =$$
$$= argmax_S \Pi_{i=1..k} \mathrm{P}(y_1|s_i, s_{i-1})\mathrm{P}(s_i|s_{i-1})$$

The Crucial Observation

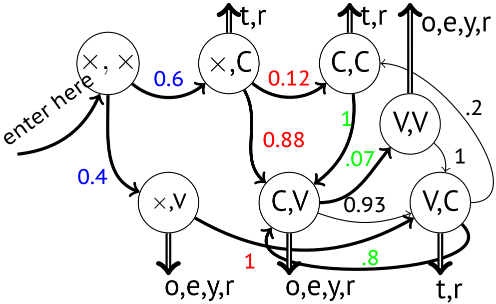
- Imagine the trellis build as before (but do not compute the αs yet; assume they are o.k.); stage *i*:



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?):



p(t|C) = .3
p(r|C) = .7
p(o|V) = .1
p(e|V) = .3
p(y|V) = .4
p(r|V) = .2

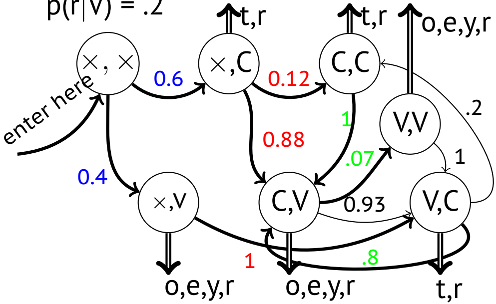
argmax_{XYZ} p(rry|XYZ) = ?

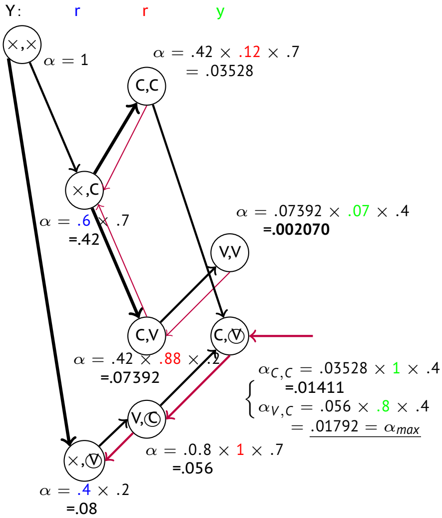
Possible state seq.: $(\times, V)(V, C)(C, V)[VCV]$, $(\times, C)(C, C)(C, V)[CCV]$, $(\times, C)(C, V)(V, V)[CVV]$

Viterbi Computation

p(t|C) = .3
p(r|C) = .7
p(o|V) = .1
p(e|V) = .3
p(y|V) = .4
p(r|V) = .2

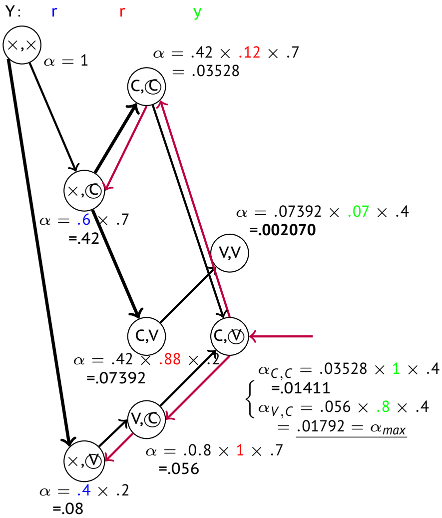
α in trellis state: best prob from start to here





n-best State Sequences

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

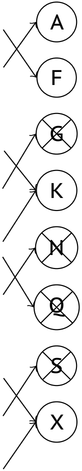


Tracking Back the n-best paths

- 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:



$\alpha = .002$

$\alpha = .043$

$\alpha = .001$

$\alpha = .231$

$\alpha = .002$

$\alpha = .000003$

$\alpha = .000435$

$\alpha = .0066$

criteria: (a) $\alpha <$ threshold
(b) $\Sigma\pi <$ threshold
(c) # of states $>$ threshold
(get rid of smallest α)