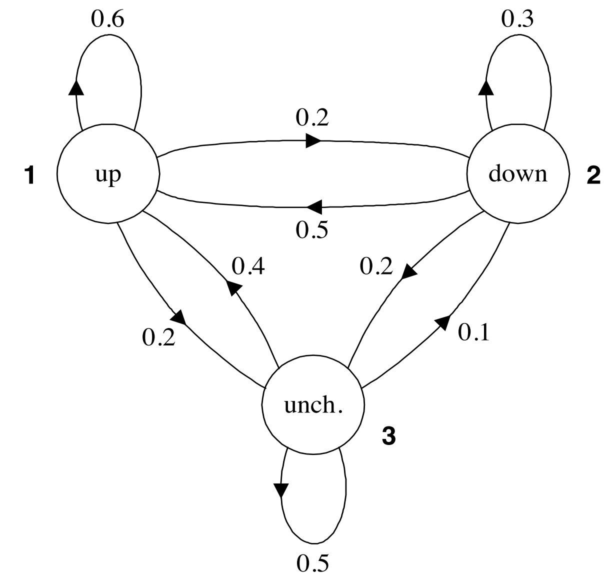


COMP90042 LECTURE 4

SEQUENCE TAGGING: HIDDEN MARKOV MODELS

OVERVIEW

- Tagging sequences
  - modelling concepts
  - Markov chains and hidden Markov models (HMMs)
  - decoding: finding best tag sequence
- Application to POS tagging

SEQUENTIAL PREDICTION

- Aim to predict *sequence* of labels for a sentence
  - instance of more general '*structured prediction*' which includes parsing
- Tagging common in NLP, e.g., part of speech
  - **Input:** *I see a silhouette of a man.*
  - **Output:** *PRP VBZ DT NN PP DT NN .*
- Other popular sequence tagging tasks include *named entity*, *shallow parsing (chunking)*
- How to build a classifier over sequences, which might be of differing lengths?

NAÏVE APPROACHES FOR SEQ. PRED.

1. Treat as one big classification label:
   - e.g., **t** = *PRP_VBZ_DT_NN_PP_DT_NN_.*
   - but there are exponentially many combinations, $|\text{Tags}|^M$ for input of length M (too many parameters!)
   - and how to tag sequences of differing lengths?

2. As *independent* classification problems
   - ***I*** *see a silhouette of a man.* → t_1 = PRP
   - *I* ***see*** *a silhouette of a man.* → t_2 = VBZ …
   - much simpler model & # parameters
   - most useful information are the neighbouring tags (but *can't be used*)

Notation key:

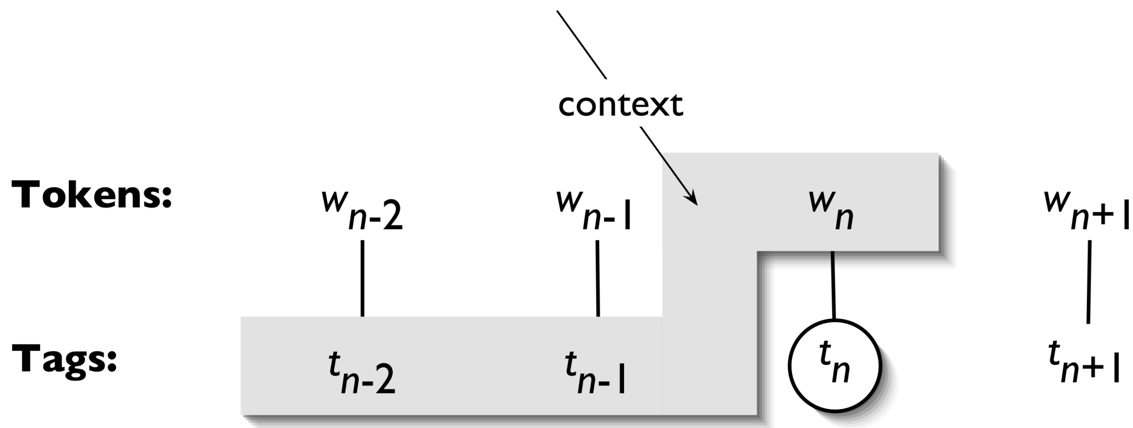
Bold (**t**) means *vector* of several values

Normal (t) means one value

Capital (A) means table (matrix) of values

A BETTER APPROACH TO SEQ. PRED.

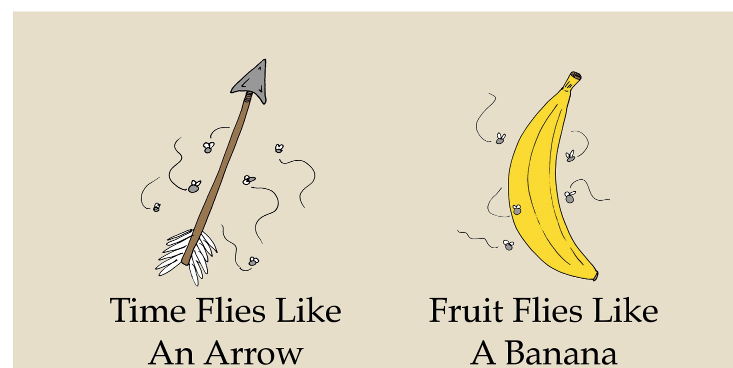
- A solution: learning a local classifier
  - e.g., $\Pr(t_n \mid w_n, t_{n-2}, t_{n-1})$ or $P(w_n, t_n \mid t_{n-1})$
  - a more practical model, has key *local* information available



- But raises *search* problem: how to find the best tag sequence for each word
  - can we avoid the exponential complexity in search?

FEATURES FOR POS TAGGING

- For part-of-speech tagging, most important features are:
  - current word
  - tags for adjacent words *(why just to the left?)*
- E.g., for *Time **flies** like an arrow*;
  - *flies* mostly seen as either a VERB or NOUN
  - *flies* is likely a VERB if previous tag is NOUN
  - *flies* is likely a NOUN if previous tag is an ADJECTIVE



http://likesuccess.com/img4568782

MARKOV CHAINS

- Useful trick to decompose complex chain of events into simpler, smaller modellable events; e.g.,
  - $\Pr(t_1\ t_2 \ldots t_n \mid w_1\ w_2 \ldots w_n) =$
    $\Pr(t_1 \mid w_1)\ \Pr(t_2 \mid w_2\ t_1) \ldots \Pr(t_n \mid w_n\ t_{n-1})$ MEMM
  - $\Pr(t_1\ t_2 \ldots t_n\ w_1\ w_2 \ldots w_n) =$
    $\Pr(t_1\ w_1)\ \Pr(t_2\ w_2 \mid t_1) \ldots \Pr(t_n\ w_n \mid t_{n-1})$ HMM
- Make some simplifying assumptions
  - *Markov assumption*: Only fixed number *k* of recent tags are relevant (*k* is known as the *Markov order;* in above *k=1*)
  - *Limited dependency between words and their tags*: Tags are assumed to capture the local context needed to explain the observations

MARKOV MODELS

- Characterised by
  - set of states
  - initial state occ prob
  - state transition probs
  - outgoing edges normalised
- Can score sequences of observations
  - For stock price example:
    up-up-down-up-up
  - maps directly to states
  - simply multiply probs

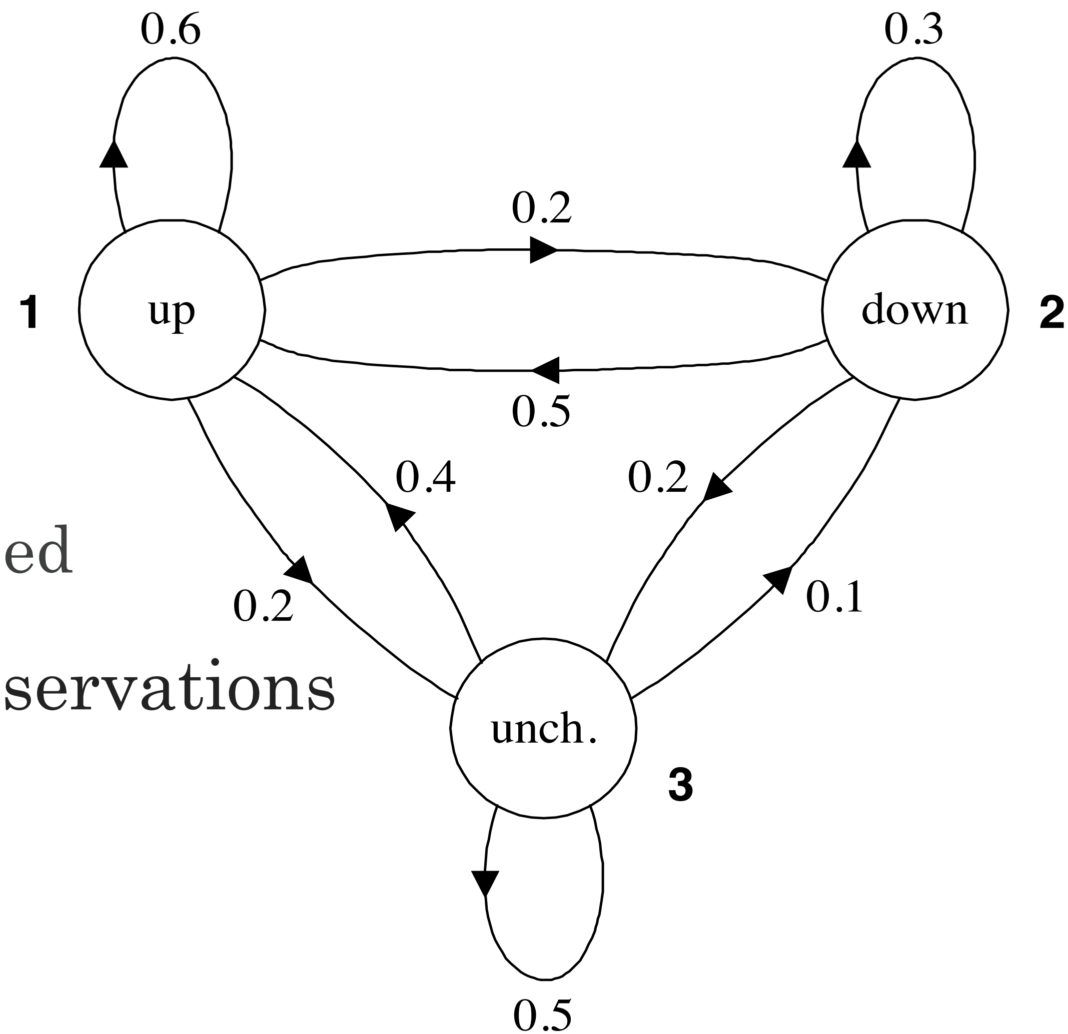


Fig. from Spoken language processing; Huang, Acero, Hon (2001); Prentice Hall

HIDDEN MARKOV MODELS

- Each state now has in addition
  - emission prob vector
- No longer 1:1 mapping
  - from observation sequence to states
  - E.g., up-up-down-up-up could be generated from any state sequence
  - but some more likely than others!
- State sequence is ‘hidden’

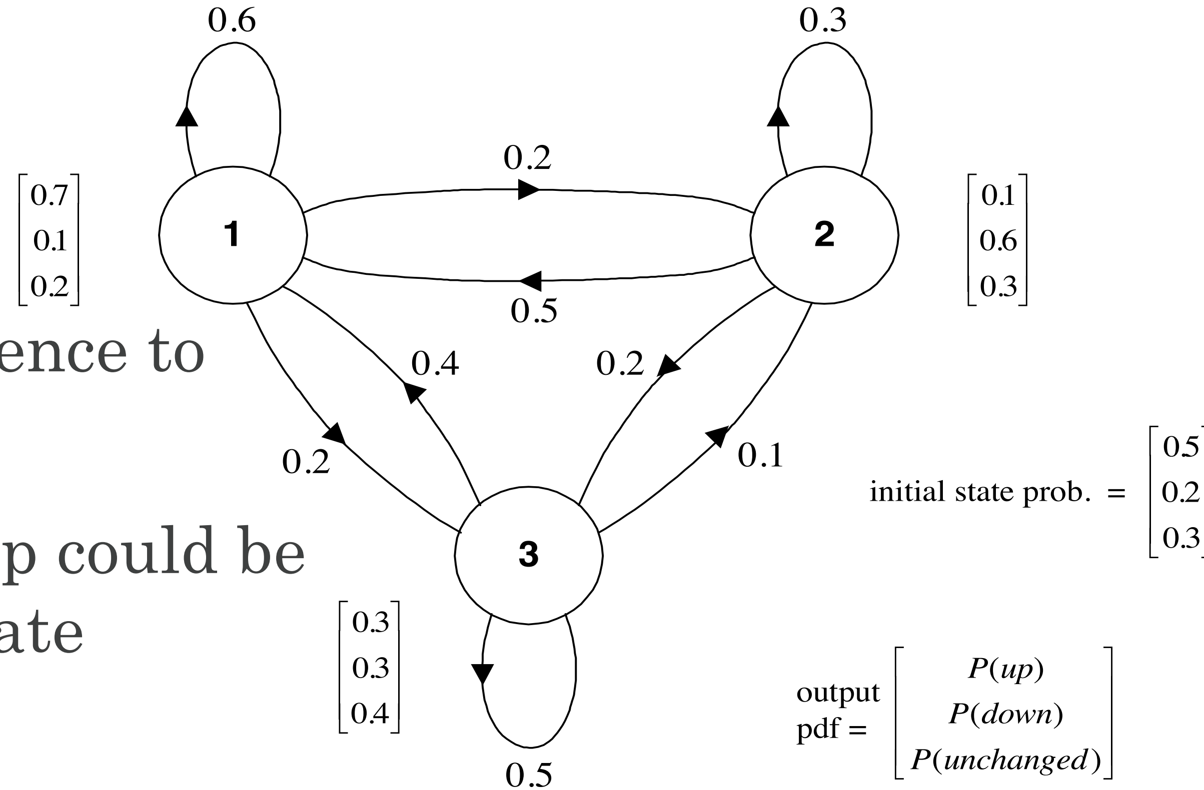


Fig. from Spoken language processing; Huang, Acero, Hon (2001); Prentice Hall

NOTATION

- Basic units are a sequence of
  - O, observations e.g., words
  - Ω, states e.g., POS tags
- Model characterised by
  - initial state probs π = vector of $|\Omega|$ elements
  - transition probs A = matrix of $|\Omega|$ x $|\Omega|$
  - emission probs O = matrix of $|\Omega|$ x $|O|$
- Together define the probability of a sequence
  - of observations together with their tags
  - a model of P(w, t)
- Notation: w = observations; t = tags; i = time index

ASSUMPTIONS

- Two assumptions underlying the HMM
- Markov assumption
  - states independent of all but most recent state
  - $P(t_i \mid t_1, t_2, t_3, \ldots, t_{i-2}, t_{i-1}) = P(t_i \mid t_{i-1})$
  - i.e., state sequence is a *Markov chain*
- Output independence
  - outputs dependent only on matching state
  - $P(w_i \mid w_1, t_1, \ldots, w_{i-1}, t_{i-1}, t_i) = P(w_i \mid t_i)$
  - forces the state t_i to carry all information linking w_i with neighbours
- Are these assumptions realistic?

PROBABILITY OF SEQUENCE

- Probability of sequence “up-up-down”

State seq	up π	O	up A	O	down A	O	total
1,1,1	0.5 x	0.7 x	0.6 x	0.7 x	0.6 x	0.1 =	0.00882
1,1,2	0.5 x	0.7 x	0.6 x	0.7 x	0.2 x	0.6 =	0.01764
1,1,3	0.5 x	0.7 x	0.6 x	0.7 x	0.2 x	0.3 =	0.00882
1,2,1	0.5 x	0.7 x	0.2 x	0.1 x	0.5 x	0.1 =	0.00035
1,2,2	0.5 x	0.7 x	0.2 x	0.1 x	0.3 x	0.6 =	0.00126
…							
3,3,3	0.3 x	0.3 x	0.5 x	0.3 x	0.5 x	0.3 =	0.00203

- 1,1,2 is the highest prob hidden sequence
- total prob is 0.054398, not 1 … why??

HMM DECODING PROBLEM

- Given observation sequence(s)
  - e.g., up-up-down-up-up
- Raises inference problem
  - what states were used to create this sequence?
  - *Viterbi* algorithm solves for the most likely states, also called '*decoding*'
- Other key inference problem (**not covered here**!)
  - unsupervised estimation where training labels are *hidden*, uses variant of the Expectation Maximisation (EM) algorithm

HMMS FOR TAGGING

- Recall part-of-speech tagging
  - time/Noun flies/Verb like/Prep an/Art arrow/Noun
- What are the units?
  - words = observations
  - tags = states
- Key challenges
  - estimate model from state-supervised data e.g., based on frequencies
  - decoding for full tag sequences

EXAMPLE

- time/Noun flies/Verb like/Prep an/Art arrow/Noun
  - Prob = P(Noun) P(time | Noun) ×
    P(Verb | Noun) P(flies | Verb) ×
    P(Prep | Verb) P(like | Prep) ×
    P(Art | Prep) P(an | Art) ×
    P(Noun | Art) P(arrow | Noun)
- time/Noun flies/Noun like/Verb an/Art arrow/Noun
  - Prob = P(Noun) P(time | Noun) ×
    P(Noun | Noun) P(flies | Noun) ×
    P(Verb | Noun) P(like | Verb) ×
    P(Art | Prep) P(an | Art) ×
    P(Noun | Art) P(arrow | Noun)
- Which do you think is more likely?
- What does a state of the art tagger choose?
  http://nlp.stanford.edu:8080/corenlp/process

ESTIMATING A VISIBLE MARKOV TAGGER

- Estimation
  - what values to use for P(w | t)?
  - what values to use for $P(t_i \mid t_{i-1})$ and $P(t_1)$?
  - use simple frequencies over training corpus, i.e.,

$$P(t_i|t_{i-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})} \qquad P(w_i|t_i) = \frac{c(w_i, t_i)}{c(t_i)}$$

E.g.,

$A_{\text{verb,noun}}$ = how often does Verb follow Noun, versus other tags?

$O_{\text{noun,`flies'}}$ = how often are Nouns written as “flies”, versus other word types?

  - (probably want to smooth counts, e.g., adding 0.5)

PREDICTION

- Prediction
  - given a sentence, **w**, find the sequence of tags, **t**

$$\arg\max_{\mathbf{t}} P(\mathbf{w}, \mathbf{t}) = P(t_1)P(w_1|t_1)\prod_{i=2}^{M} P(t_i|t_{i-1})P(w_i|t_i)$$

$$= \pi_{t_1} O_{t_1,w_1} \prod_{i=2}^{M} A_{t_{i-1},t_i} O_{t_i,w_i}$$

  - problems
    - exponential number of values of t
    - but computation can be factorised…

VITERBI ALGORITHM

- Form of dynamic programming to solve maximisation
  - define matrix α of size M (length) x T (tags)

$$\alpha[i, t_i] = \max_{t_1 \cdots t_{i-1}} P(w_1 \cdots w_i, t_1 \cdots t_i)$$

  - full sequence max is then

$$\max_{\vec{t}} P(\vec{w}, \vec{t}) = \max_{t_M} \alpha[M, t_M]$$

  - how to compute α?
- We're interested in the **tags**, not just the **max** value
  - can be recovered from α using back-pointers

VITERBI RECURSION

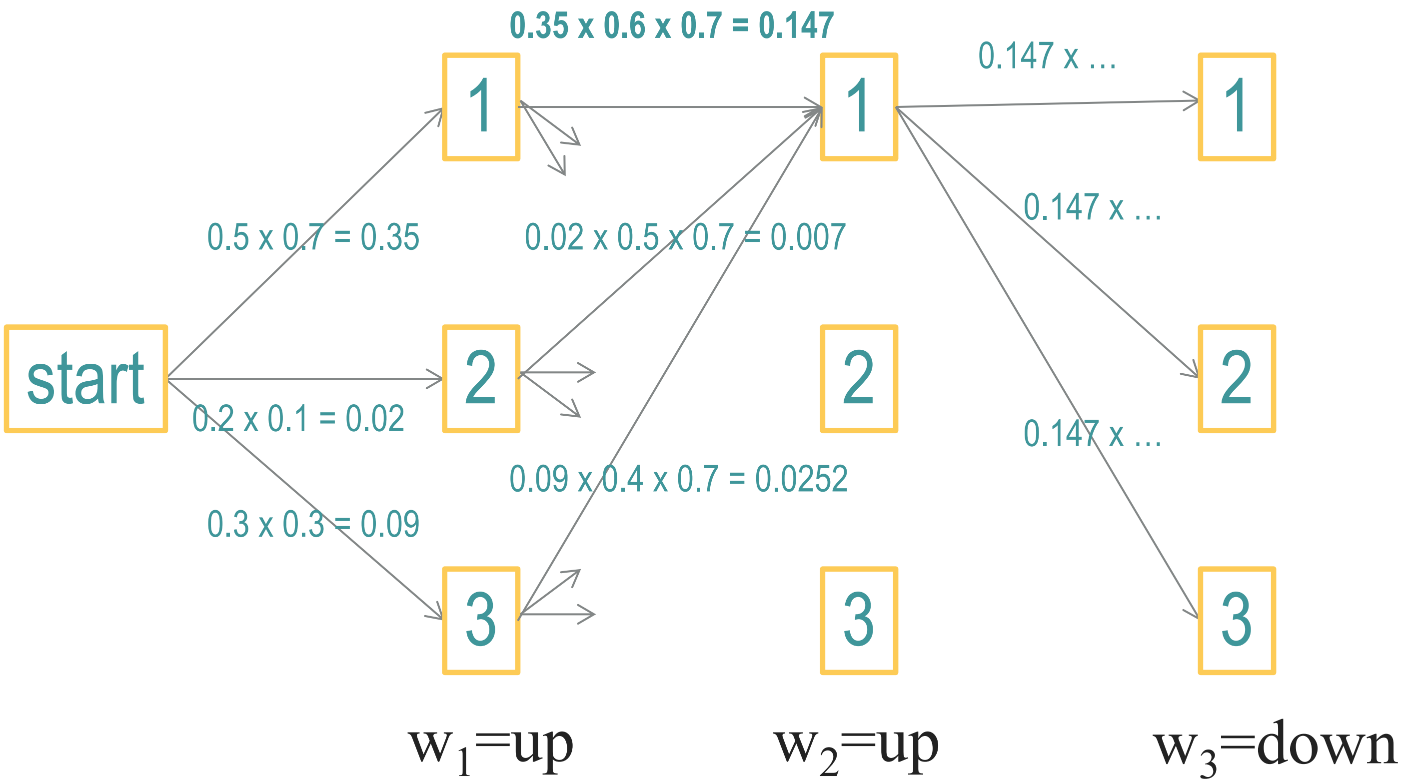
- Can be defined recursively

$$\begin{aligned}\alpha[i, t_i] &= \max_{t_1 \cdots t_{i-1}} P(w_1 \cdots w_i, t_1 \cdots t_i) \\ &= \max_{t_1} \cdots \max_{t_{i-2}} \max_{t_{i-1}} P(w_1 \cdots w_i, t_1 \cdots t_i) \\ &= \max_{t_1} \cdots \max_{t_{i-2}} \max_{t_{i-1}} P(w_1 \cdots w_{i-1}, t_1 \cdots t_{i-1}) P(w_i, t_i | t_{i-1}) \\ &= \max_{t_{i-1}} \alpha[i-1, t_{i-1}] P(w_i, t_i | t_{i-1})\end{aligned}$$

- Need a base case to terminate recursion

$$\alpha[1, t_1] = P(w_1, t_1)$$

VITERBI ILLUSTRATION



- All maximising sequences with t_2=1 must also have t_1=1
- No need to consider extending [2,1] or [3,1].

VITERBI ANALYSIS

- Algorithm as follows

```
alpha = np.zeros(M, T)
for t in range(T):
    alpha[1, t] = pi[t] * O[w[1], t]

for i in range(2, M):
    for t_i in range(T):
        for t_last in range(T):    # t_last means t_{i-1}
            alpha[i,t_i] = np.max(
                alpha[i,t_i],
                alpha[i-1, t_last] * A[t_last, t_i] * O[w[i], t_i])

best = np.max(alpha[M-1,:])
```

- Time complexity is $O(M\ T^2)$

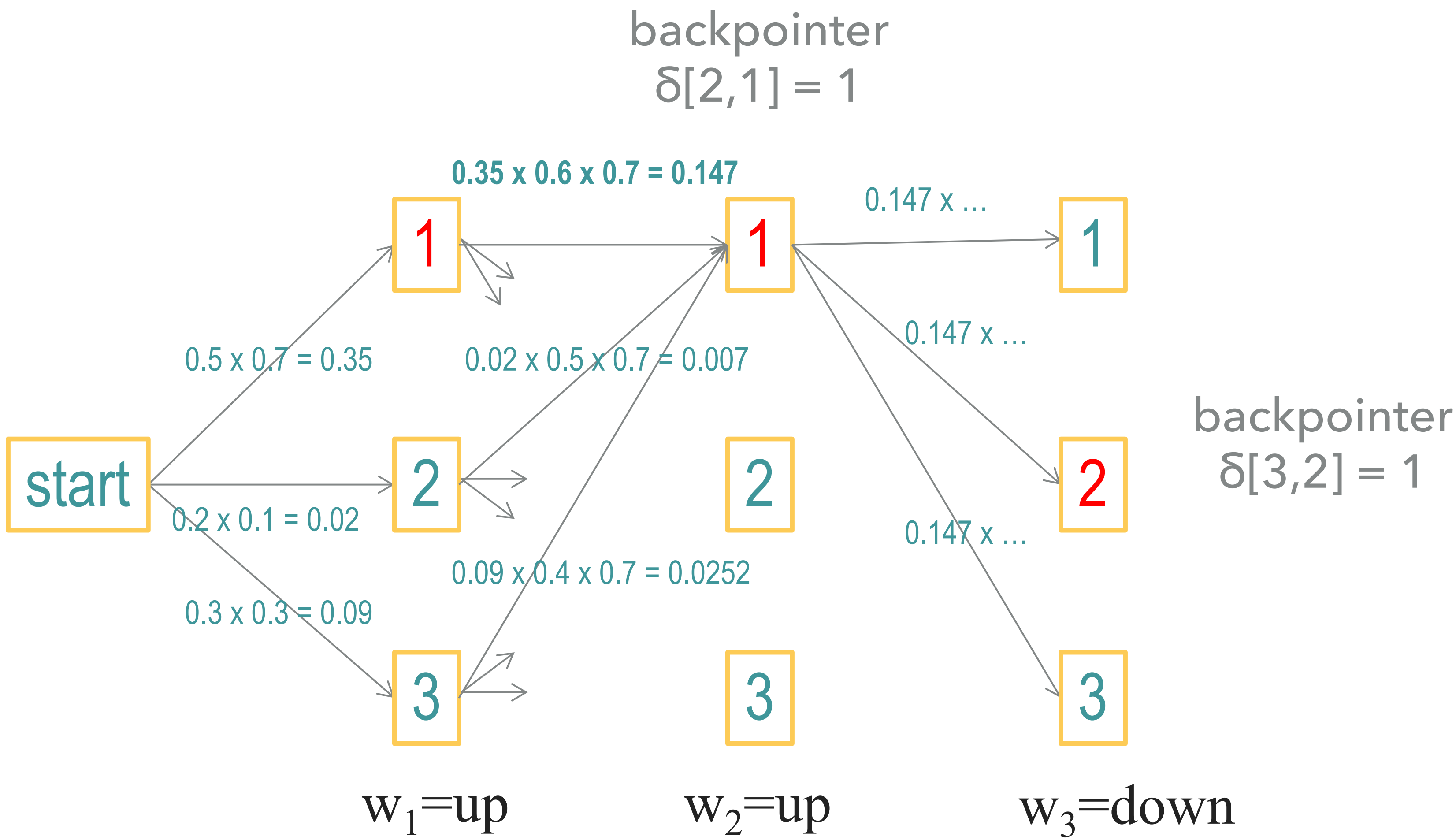
BACKPOINTERS

- Know the maximum score, but not the best path of states
- Solution: don't just store *max* values, α, but who '*won*' each maximisation (the *arg max*), i.e.,

```
for i in range(2, M):
    for t_i in range(T):
        for t_last in range(T):    # t_last means t_{i-1}
            s = alpha[i-1, t_last] * A[t_last, t_i] * O[w[i], t_i]
            if s >= alpha[i,t_i]:
                alpha[i,t_i] = s
                back[i,t_i] = t_last     # record best so far
```

- At the end, must travel backwards from last tag to first to read off the reverse tag sequence

BACKPOINTER ILLUSTRATION



OTHER VARIANT TAGGERS

- HMM is **generative**, $P(\boldsymbol{t}, \boldsymbol{w})$, 'creates' the input
  - allows for unsupervised HMMs: learn model without any tagged data!
- **Discriminative** models also popular, modelling $P(\boldsymbol{t} \mid \boldsymbol{w})$ directly
  - supports richer feature set, generally better accuracy when trained over large supervised datasets
  - E.g., Maximum Entropy Markov Model (MEMM), Conditional random field (CRF), Connectionist Temporal Classification (CTC)
  - Most *deep learning* models of sequences are discriminative (e.g., encoder-decoders for translation), similar to an MEMM

HMMS IN NLP

- HMMs are highly effective for part-of-speech tagging
  - trigram HMM gets 96.5% accuracy (TnT)
  - related models are state of the art
    - MEMMs 97.3%
    - CRFs 97.6%
  - *English Penn Treebank* tagging accuracy
    https://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)
- Other sequence labelling tasks
  - named entity recognition, shallow parsing, alignment …
  - In other fields: DNA, protein sequences, image lattices…

SUMMARY

- Probabilistic models of sequences
- Introduced hidden Markov models
  - supervised estimation for learning
  - Viterbi algorithm for efficient prediction
  - relationship to other discriminative sequence models

READINGS

- Hidden Markov models
  - Jurafsky & Martin 2nd Ed., chapter 6
- [Optional] Rabiner's HMM tutorial, for more details
  - http://tinyurl.com/2hqaf8
- **[Just for fun!]** Contemporary sequence tagging methods
  - Lafferty et al, Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001), ICML