11-411/11-611 Natural Language Processing

Sequence Labeling

David R. Mortensen

February 23, 2023

Language Technologies Institute

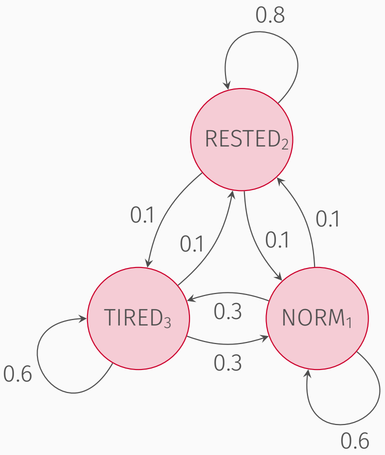
Learning Objectives

At the end of this lecture, you should be able to do the following things:

- Describe HMMs and their relationship to Markov Chains
- Implement the Forward Algorithm (and walk through it with pencil and paper)
- Implement the Viterbi Algorithm (and walk through it with pencil and paper)
- Be able to state the basic properties of Conditional Random Fields
- Be able to describe how RNNs are used for sequence labeling

Markov Chains

Markov Chains Tell Us about the Probabilities of Sequences of Random Variables



The figure to the left represents a Markov Chain.

- States
- Transitions
- Weights (probabilities)

The probability of RESTED$_2$ at the timestep after RESTED$_2$ is **0.8**.
The probability of NORM$_1$ after RESTED$_2$ is **0.1**.
The probability of NORM$_1$ $\rightarrow$ TIRED$_3$ $\rightarrow$ RESTED$_2$ is $0.3 \times 0.1 = \mathbf{0.03}$
The Markov Assumption applies.

The Markov Assumption

"When predicting the future, the past doesn't matter—only the present."

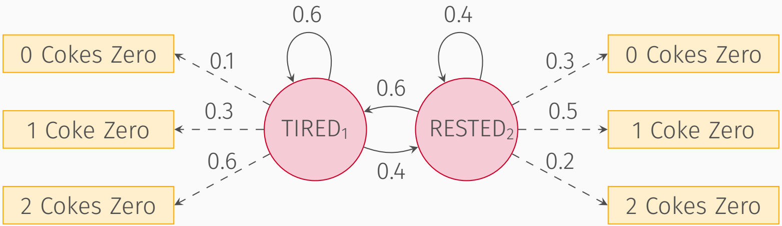
In other words

$$p(q_i = a|q_1 \ldots q_{i-1}) = P(q_i = a|q_{i-1}) \tag{1}$$

This is the same assumption we made for ngram language modeling.

Hidden Markov Models: the Coke Zero Example

Since I do not drink coffee, I must drink Coke Zero to remain caffeinated. My consumption is related to my exhaustion. Could you build a model to infer my exhaustion from the number of Coke Zero bottles added to my wastebasket each day?



$\pi = [0.7, 0.3]$ (the initial probabilities of states)

In Hidden Markov Models, Markov Chains are Hidden

The basic idea of a Hidden Markov Model (or HMM) is like that of a Markov Chain, except that the states are never observed.

- Observations are "emitted" from the hidden states
- So, when seen as a graph, HMMs have two kinds of nodes and two kinds of edges
  - Hidden states and observations
  - Transitional probabilities and emission probabilities
- The hidden states and transitional probabilities represent the latent structure (**tiredness**) that "sits behind" the observed phenomena (**Coke Zero bottles**)

A Formal Defintion of the Hidden Markov Model

$Q = q_1, \ldots, q_N$	a set of N states
$A = a_{1,1}, a_{1,2}, \ldots$	a transitional probability matrix of cells a_{ij}, where each cell is a probability of moving from state i to state j. $\sum_{j=1}^{N} a_{ij} = 1 \; \forall i$
$O = o_1, \ldots, o_T$	a sequence of T observations, each drawn from a vocabulary V.
$B = b_1, \ldots, b_n$	a sequence of observation likelihoods (or emission probabilities). The probability that observation o_t is generated by state q_i.
$\pi = \pi_1, \ldots, \pi_N$	an initial probability distribution over states (the probability that the Markov chain will start in state q_i. Some states q_j may have $p_j = 0$ (meaning they cannot be initial states). $\sum_{i=1}^{N} \pi_i = 1 \; \forall i$

HMMs Assume the Markov Assumption and Output Independence

Like Markov Chains, HMMs require the Markov Assumption:

$$P(q_i|q_1 ... q_{i-1}) \approx P(q_i|q_{i-1}) \tag{2}$$

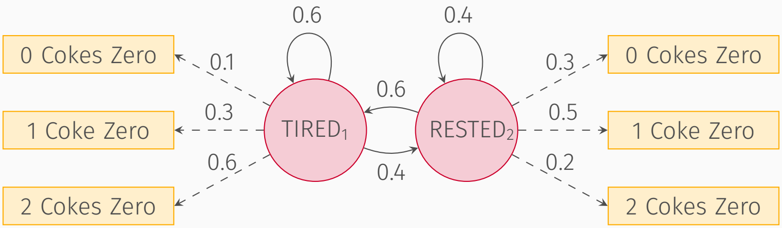
The further assume that the observed outputs depend only upon the state (**output independence**)

$$P(o_i|q_1, \ldots, q_i, \ldots, q_T, o_1 \ldots, o_i, \ldots o_T) \approx P(o_i|q_i) \tag{3}$$

Where $q_1, \ldots, q_T$ are the states at each time step and $o_1, \ldots, o_T$ are the outputs at each time step. In other words:

- The preceding or following states do not matter (we assume)
- The preceding or following outputs do not matter (we assume)

Returning to the Coke Zero Example



$\pi = [0.7, 0.3]$ (the initial probabilities of states)

The Three HMM Problems are Likelihood, Decoding, and Learning

Likelihood Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

Decoding Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.

Learning Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Computing Likelihood with the Forward Algorithm

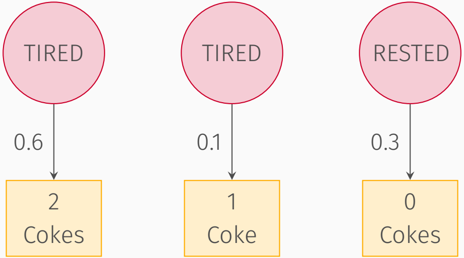
The Likelihood Problem Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

This was easy with a Markov Chain (since we could observe the states): we just followed the path and multiplied the probabilities.

For an HMM, things are not so simple. Lets start with a simpler problem.

What if We Knew Which Days David Was Tired?

To simplify things, let's start off by assuming that we actually knew the sequence of tired/rested days and wanted to predict the probability of a Coke Zero sequence. This is not hard:



$$P(2\ 1\ 0|\text{tired tired rested}) = P(0|\text{tired}) \times P(1|\text{tired}) \times P(1|\text{rested}) \tag{4}$$

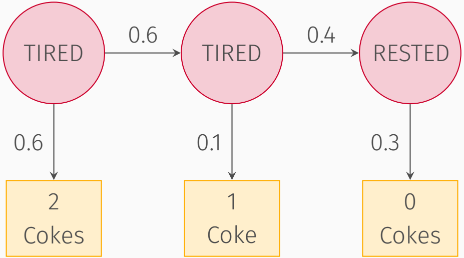
However, We Don't Know Which Days I Was Tired

We don't know the hidden sequence; we can only see the trail of Coke Zero bottles that David leaves behind.

Instead we might compute the probability of Coke events by summing over all possible sequences of tired/rested days, weighted by their probability:

$$P(O, Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} P(q_i|q_{i-1}) \tag{5}$$

Computing Joint Probability of Emission and Transition



$$P(2\ 1\ 0, \text{tired tired rested}) = (0.6 \times 0.1 \times 0.3) \times (0.6 \times 0.4) = 0.00432 \tag{6}$$

However, computing the probability of a sequence in this way seems intractable since $|Q|$ and $|O|$ are arbitrarily large.

The Forward Algorithm shows us a better way.

a_{ij} the **transition probbilities** from previous state q_i to current state q_j

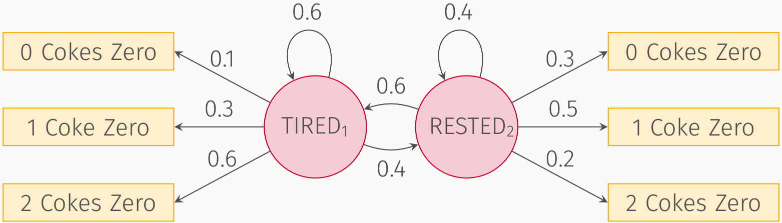
$b_j(o_t)$ the **state observation likelihood** of the observed symbol o_t given the current state j (emission probability)

The Forward Algorithm is a Dynamic Programming Algorithm

function FORWARD(observations of len T state-graph of len N) **returns** forward-prob
  create a probability matrix $forward[N, T]$
  for each s from 1 to N **do**
    $forward[s, 1] \leftarrow \pi_s * b_s(o_1)$ ▷ initialization step
  for each t from 2 to T **do**
    for each s from 1 to N **do**
      $forward[s, t] \leftarrow \sum_{s'=1}^{N} forward[s', t-1] * a_{s',s} * b_s(o_t)$ ▷ recursion step
  $forwardprob \leftarrow \sum_{s=1}^{N} forward[s, T]$ ▷ termination step
  return *forwardprob*

A dynamic programming algorithm stores values that can be used to compute other values rather than recomputing these values multiple times (in this case, in the matrix *forward*). We scan the matrix, computing a value for each cell based on values we have computed before (recursion step).

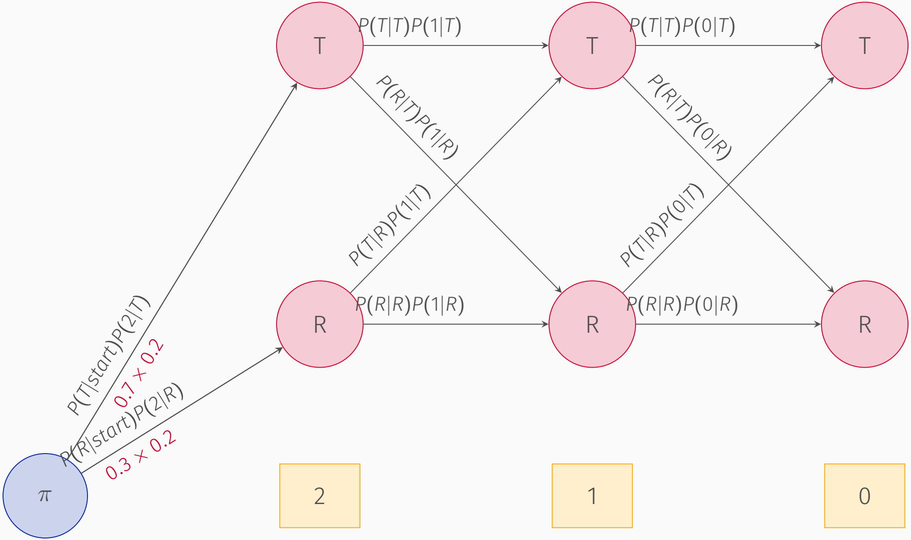
A Reminder: Here Is Our HMM



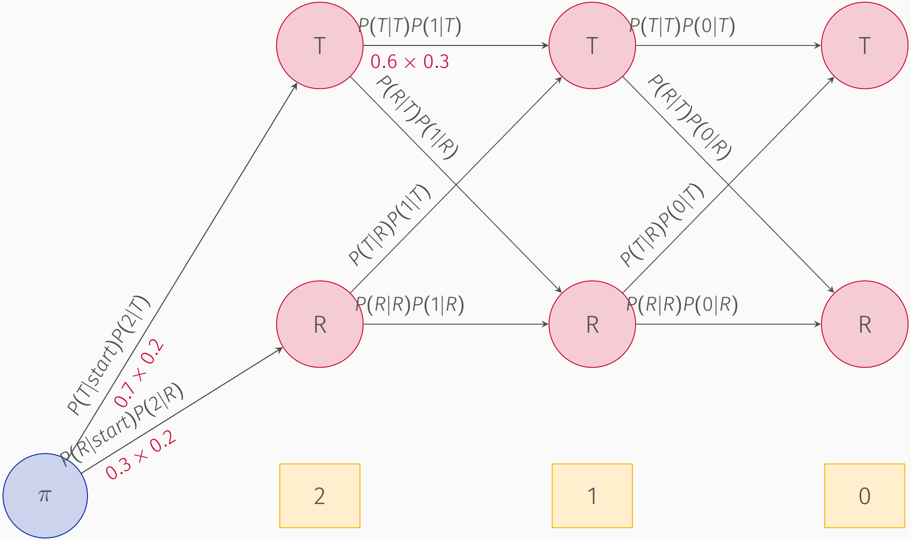
$$\pi = [0.7, 0.3]$$

Our sequence of observations is [2, 1, 0].

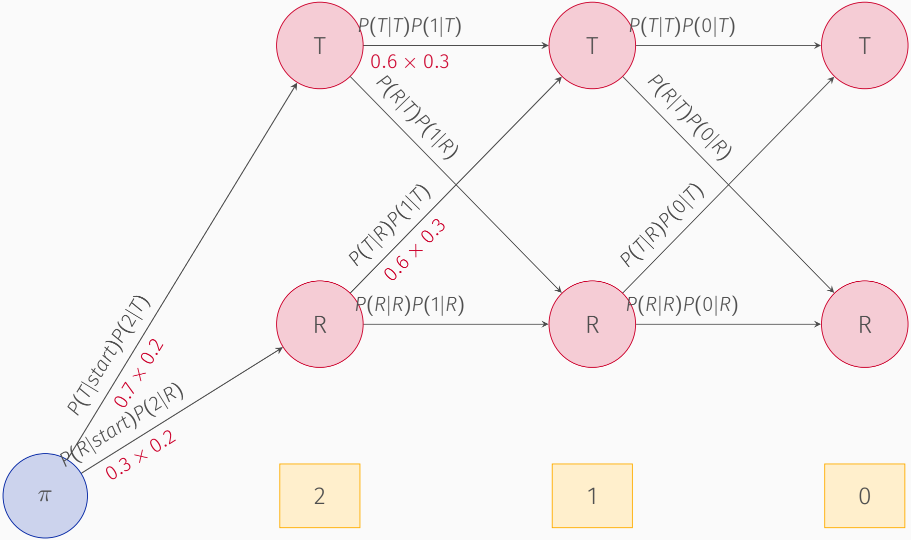
Computing a Forward Trellis



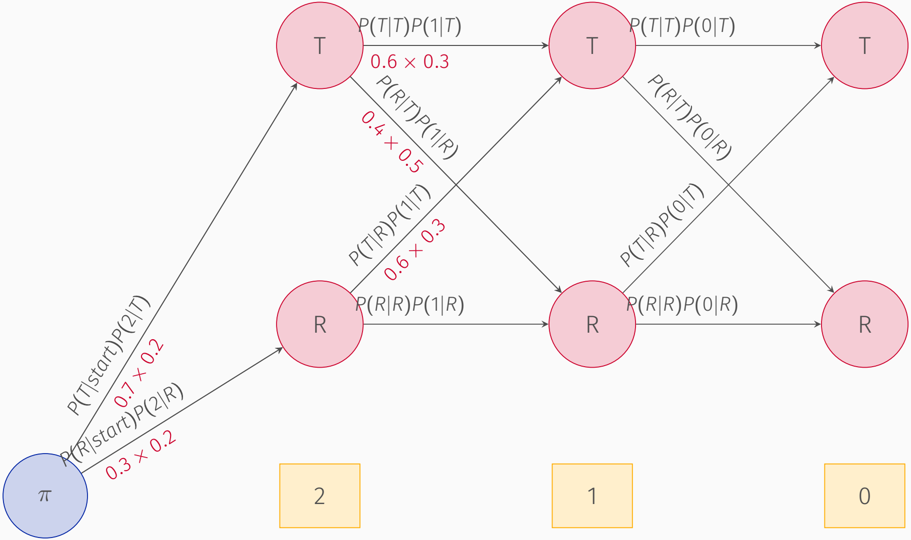
Computing a Forward Trellis



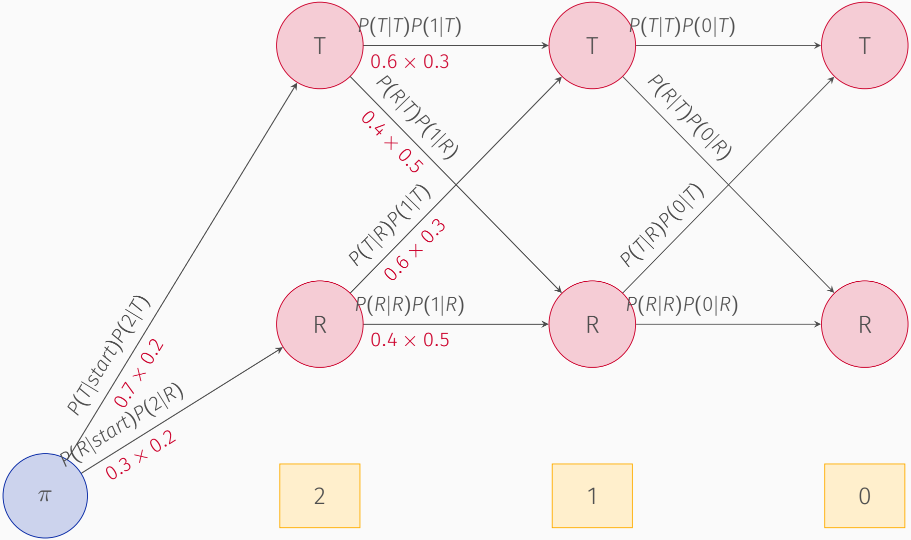
Computing a Forward Trellis



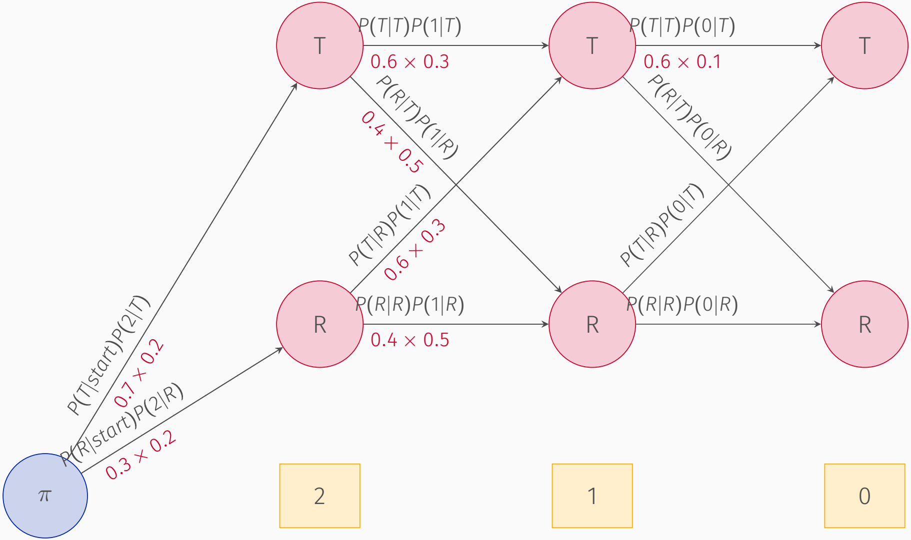
Computing a Forward Trellis



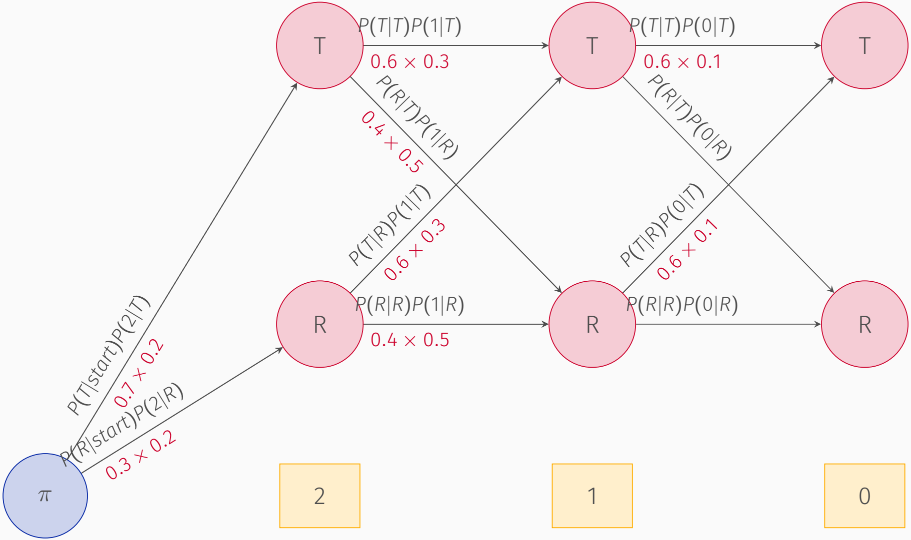
Computing a Forward Trellis



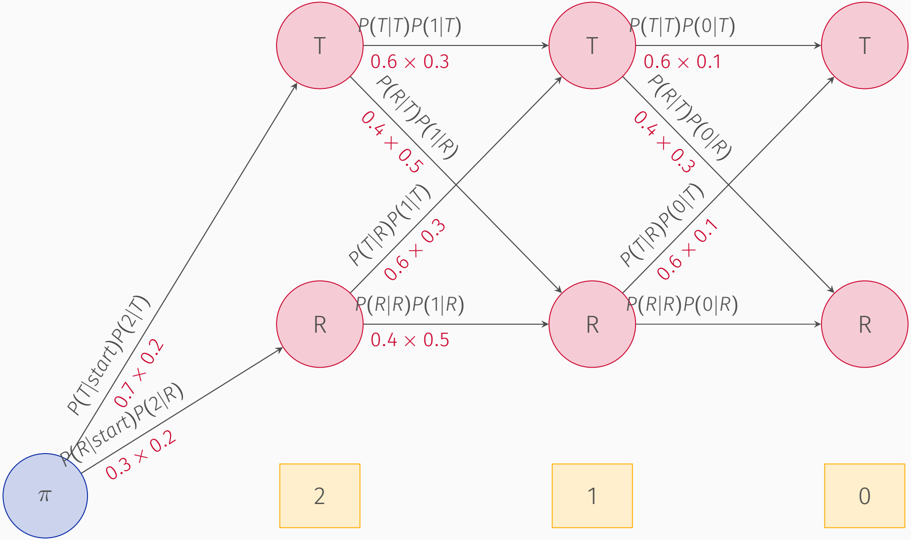
Computing a Forward Trellis



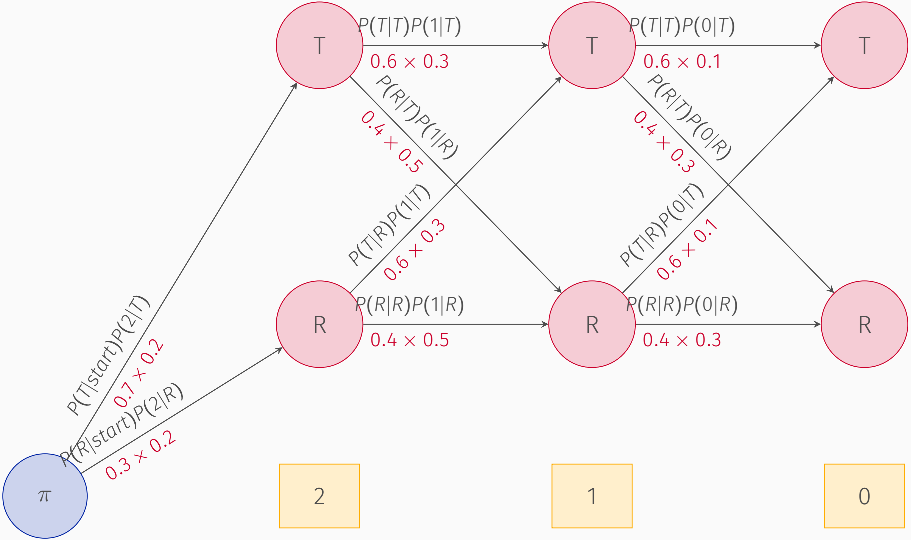
Computing a Forward Trellis



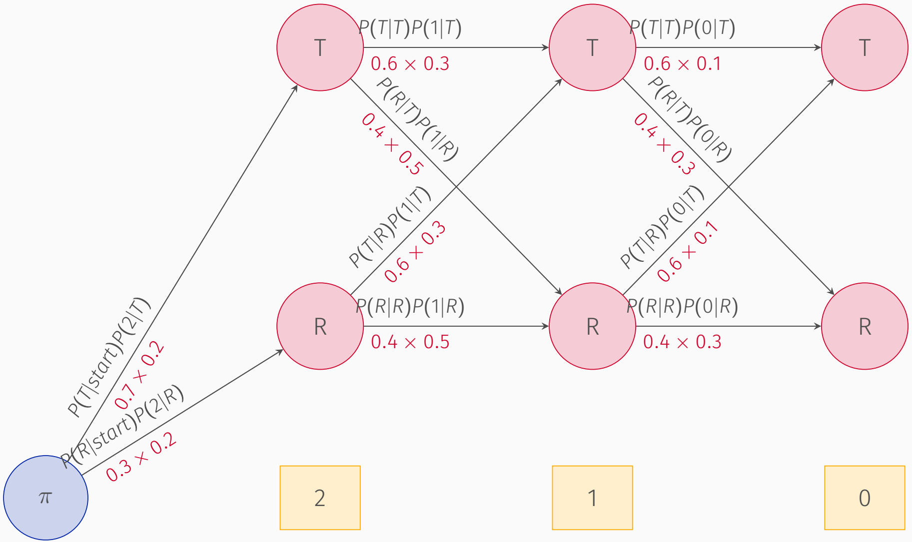
Computing a Forward Trellis



Computing a Forward Trellis



Computing a Forward Trellis



Often, We Want to Decode HMMs

Input A trained HMM and a series of observations

Output A series of labels, corresponding to hidden states of the HMM

This task shows up many times:

- Labeling words according to their parts of speech
- Labeling words according to whether they are at the beginning, otherwise inside of, or outside of a name
- Inferring the sequence of tired and not tired days in the month of your instructor based on his Coke Zero consumption

This is Decoding

More formally, given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = 0_1 o_2, \ldots, o_T$, find the most probable sequence of states $Q = q_1 q_2, \ldots, q_T$.

Can We Decode with the Forward Algorithm?

In principle, we could use the Forward Algorithm for decoding, but we would have to compute the probability for all possible sequences of states.

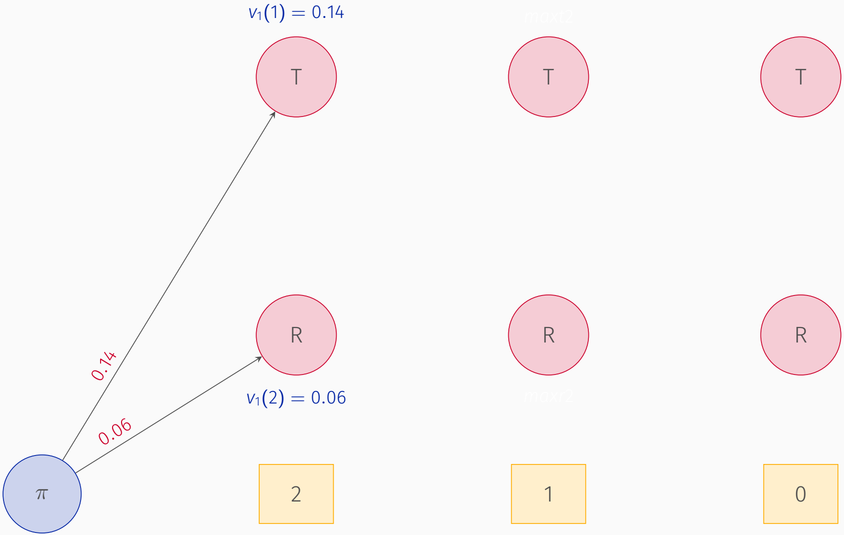
This is computationally infeasible because the set of possible state sequences (e.g. TTT, TRT, TRR, RRR, ...) grows exponentially as the number of states N grows.

Fortunately, there is another way.

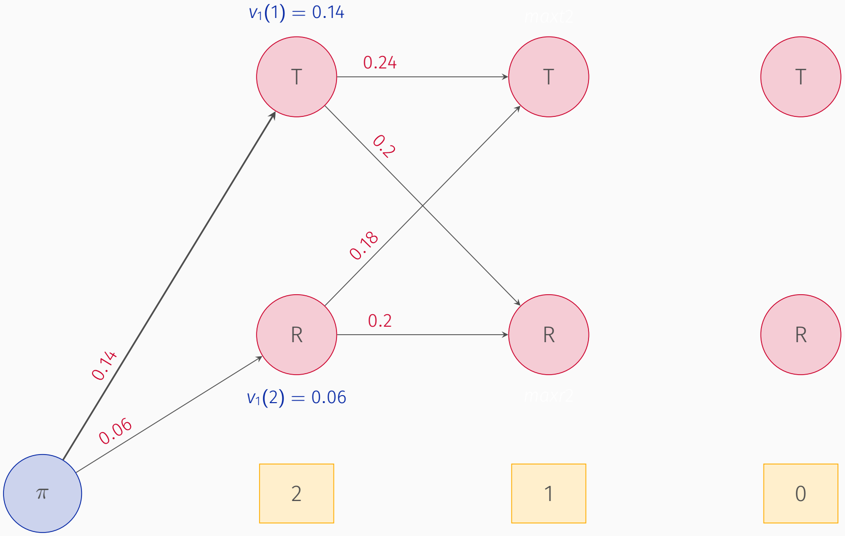
The Viterbi Algorithm Can Be Used to Decode HMMs

function $\textsc{Viterbi}$(observations $O = o_1, o_2, \ldots, 0_T$, state-graph of length N)
  $V[N, T] \leftarrow$ empty path probability matrix
  $B[N, T] \leftarrow$ empty backpointer matrix
  for each $s \in 1..N$ **do**
    $V[s, 1] \leftarrow \pi_s \cdot b_s(o_1)$
    $B[s, 1] \leftarrow 0$
  for each $t \in 2..T$ **do**
    for each $s \in 1..N$ **do**
      $V[s, t] \leftarrow \max_{s'=1}^{N} V[s', t-1] \cdot a_{s',s} \cdot b_s(o_t)$
      $B[s, t] \leftarrow \operatorname{argmax}_{s'=1}^{N} V[s', t-1] \cdot a_{s',s} \cdot b_s(o_t)$
  $bestpathprob \leftarrow \max_{s=1}^{N} V[s, T]$
  $bestpathpointer \leftarrow \max_{s=1}^{N} V[s, T]$
  $bestpath \leftarrow$ path starting at *bestpathpointer* that follows b to states back in time.
  return $bestpath, bestpathprob$

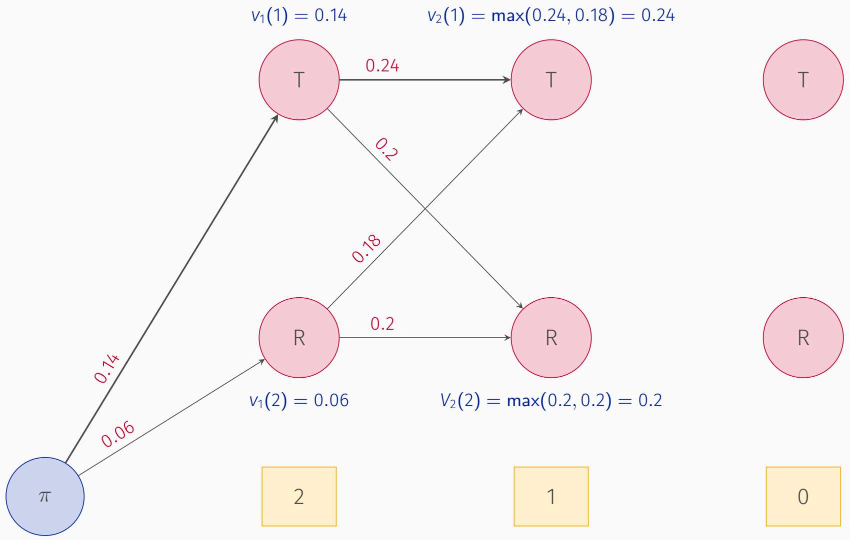
Using Viterbi to Decode an HMM



Using Viterbi to Decode an HMM



Using Viterbi to Decode an HMM



Decoding with Viterbi and Our HMM

Cumulative probabilities are represented here in log space:

(Note the the order of T and R has changed).

Decoding with Viterbi and Our HMM

Cumulative probabilities are represented here in log space:

(Note the the order of T and R has changed).

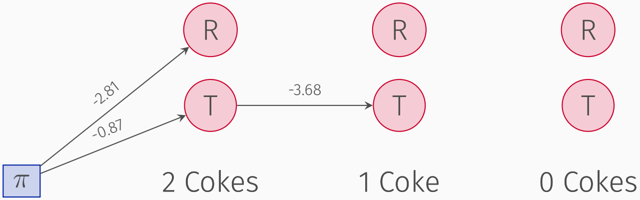
Decoding with Viterbi and Our HMM

Cumulative probabilities are represented here in log space:

(Note the the order of T and R has changed).

Decoding with Viterbi and Our HMM

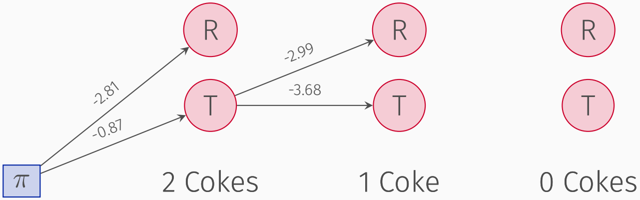
Cumulative probabilities are represented here in log space:



(Note the the order of T and R has changed).

Decoding with Viterbi and Our HMM

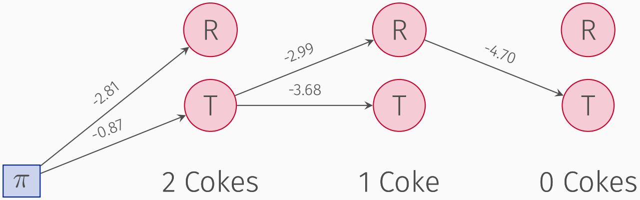
Cumulative probabilities are represented here in log space:



(Note the the order of T and R has changed).

Decoding with Viterbi and Our HMM

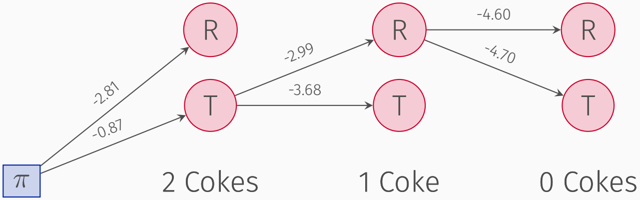
Cumulative probabilities are represented here in log space:



(Note the the order of T and R has changed).

Decoding with Viterbi and Our HMM

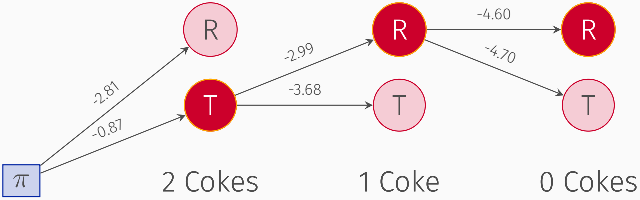
Cumulative probabilities are represented here in log space:



(Note the the order of T and R has changed).

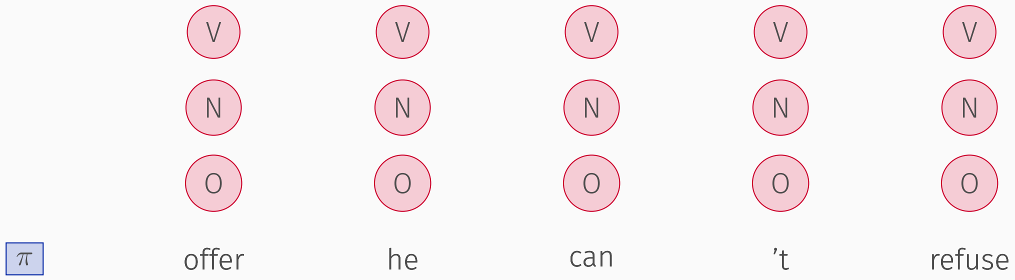
Decoding with Viterbi and Our HMM

Cumulative probabilities are represented here in log space:

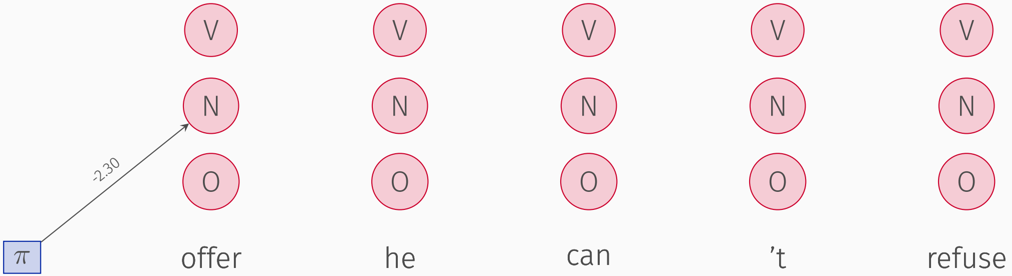


(Note the the order of T and R has changed).

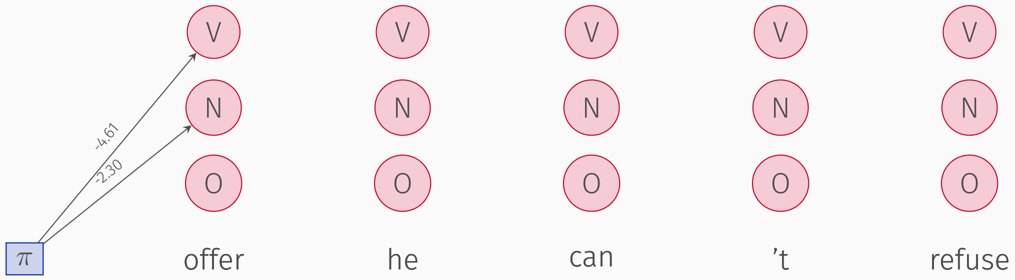
A Trellis You Can’t Refuse: Viteri for Parts of Speech



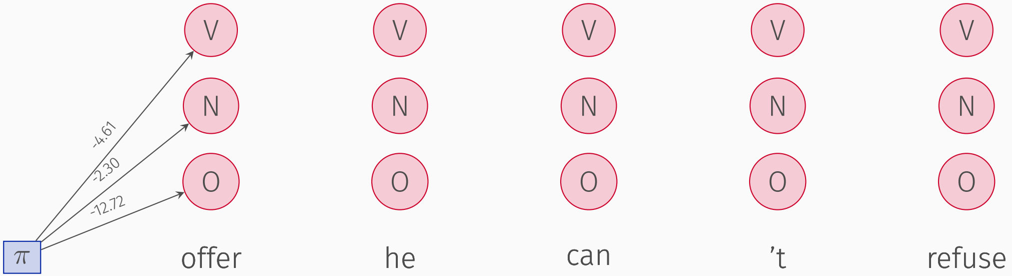
A Trellis You Can't Refuse: Viteri for Parts of Speech



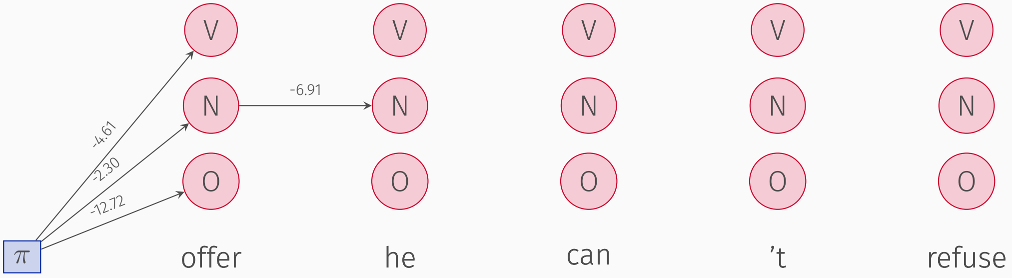
A Trellis You Can't Refuse: Viteri for Parts of Speech



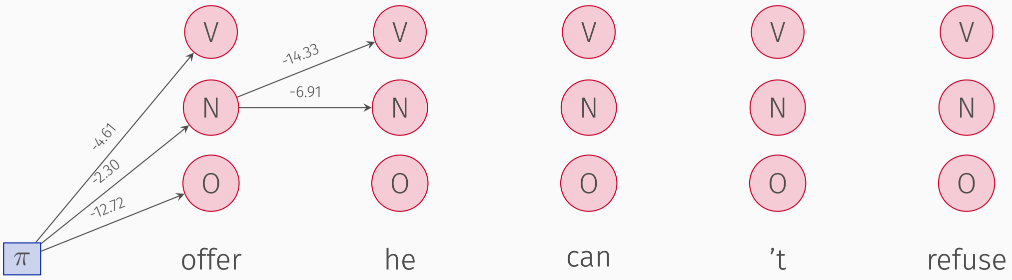
A Trellis You Can't Refuse: Viteri for Parts of Speech



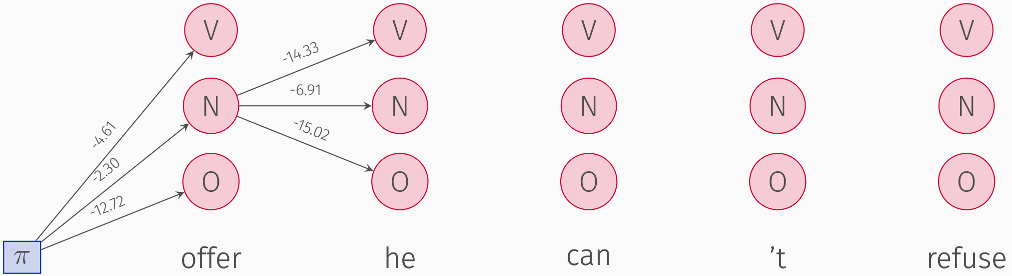
A Trellis You Can't Refuse: Viteri for Parts of Speech



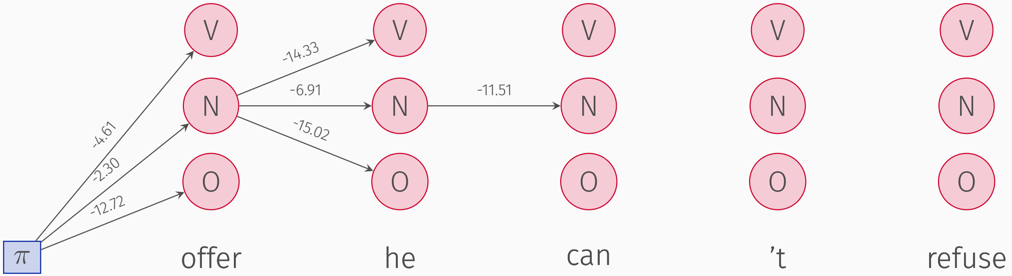
A Trellis You Can't Refuse: Viteri for Parts of Speech



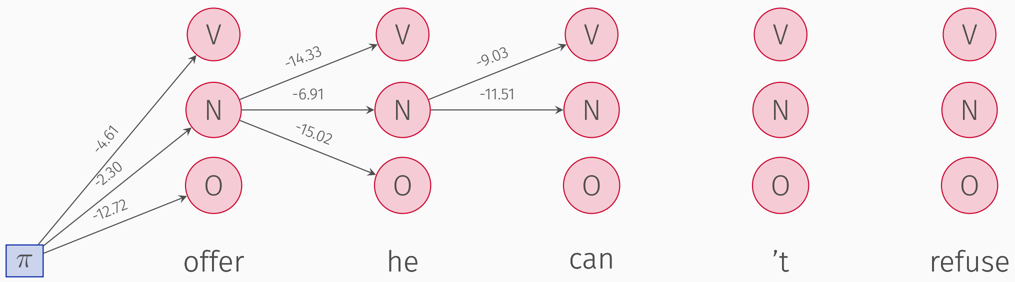
A Trellis You Can't Refuse: Viteri for Parts of Speech



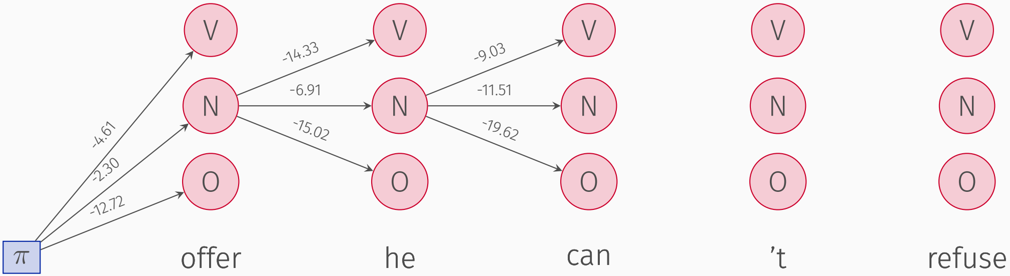
A Trellis You Can't Refuse: Viteri for Parts of Speech



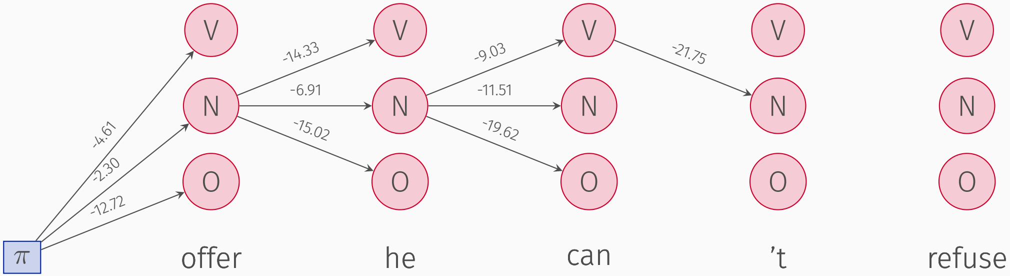
A Trellis You Can’t Refuse: Viteri for Parts of Speech



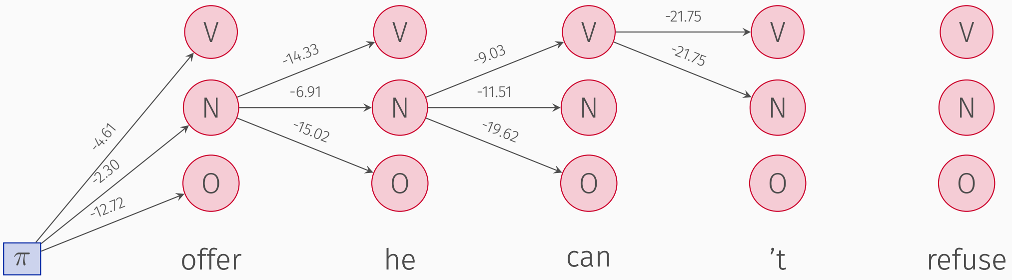
A Trellis You Can't Refuse: Viteri for Parts of Speech



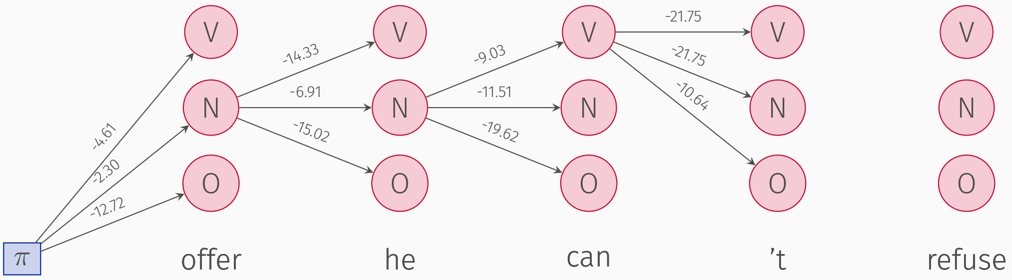
A Trellis You Can't Refuse: Viteri for Parts of Speech



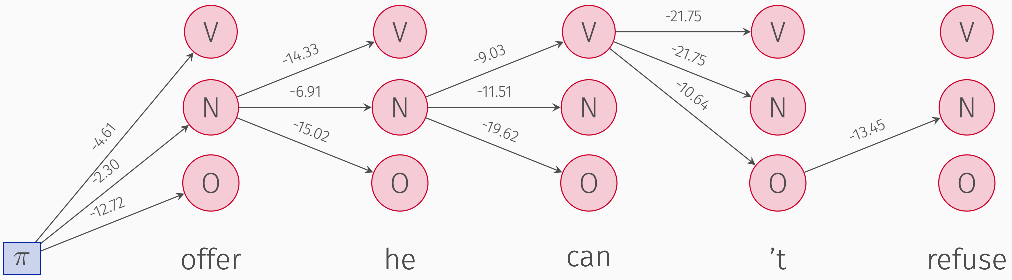
A Trellis You Can't Refuse: Viteri for Parts of Speech



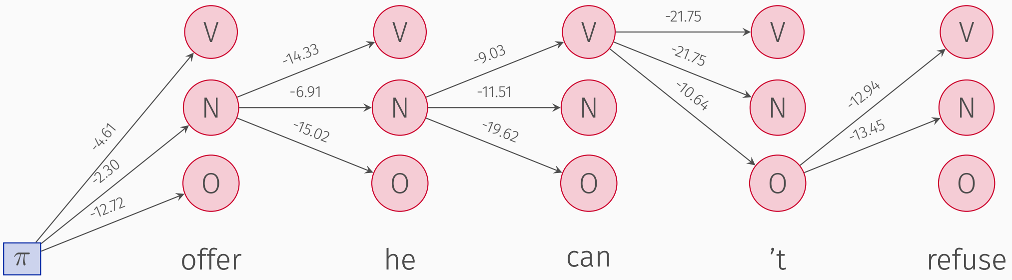
A Trellis You Can't Refuse: Viteri for Parts of Speech



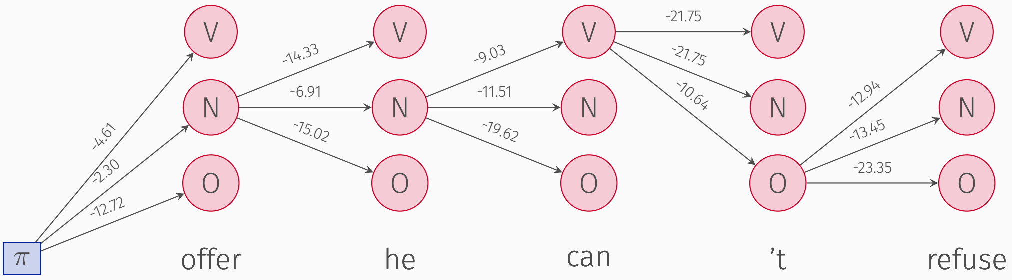
A Trellis You Can't Refuse: Viteri for Parts of Speech



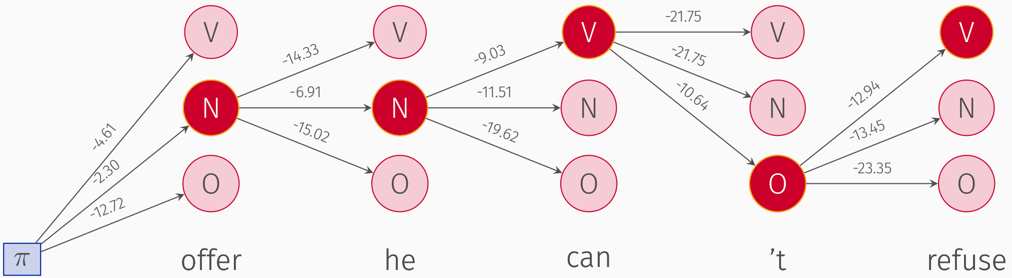
A Trellis You Can’t Refuse: Viteri for Parts of Speech



A Trellis You Can't Refuse: Viteri for Parts of Speech



A Trellis You Can't Refuse: Viteri for Parts of Speech



Conditional Random Fields

Conditional Random Fields are Also Used to Label Sequences

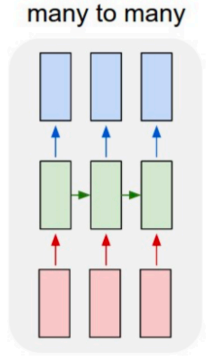
…and they work better than HMMs, but are more complicated to understand.

CRFs are a kind of discriminative **undirected** probablistic graphical model (whereas HMMs are a kind of generative directed probablistic graphical model).

CRFs are now used **much** more frequently that HMMs, sometimes in conjunction with neural networks like RNNs. I won't have time to talk much about them here, but you can understand them as playing a similar role to HMMs.

Sequence Labeling with RNNs

RNNs Can Also Be Used for Sequence Labeling



An RNN that emits an output (from specified vocabulary) for every input is essentially doing sequence labeling.
In a successful model for NER (named entity recognition, the task of labeling words based on the status as parts of names), bidirectional LSTMs are combined with CNNs (convolutional neural networks), which do feature extraction from words, and CRFs, which actually assign the labels.
We will learn more about NER (and POS tagging) in the next lecture.

Questions?