Markov jokes:

Once you’ve heard the latest one, you’ve heard them all.

# CS 2731 Introduction to Natural Language Processing

## Session 19: HMMs part 2, Viterbi algorithm, neural sequence labeling

Michael Miller Yoder

March 20, 2024

University of Pittsburgh | School of Computing and Information

# Course logistics: homeworks

- Homework 3 grades released
- Homework 4 is **due Mon Mar 25**
  - Part 1: Do part-of-speech tagging manually with the Viterbi algorithm
  - Part 2: Fine-tune BERT-based models for part-of-speech tagging in English and Norwegian
    - Copy and fill in a skeleton Colab notebook

# Course logistics: project

- Project peer review **due Wed Mar 27**
  - Was released today
  - Form where you will review your own and your teammates’ contributions so far
  - Will not be used for grading, just for addressing any issues
- Basic working system **due Thu Apr 4**

# Overview: HMMs part 2, Viterbi alg, neural sequence labeling

- HMMs review
- Training HMMs
- Decoding HMMs: Viterbi algorithm
- Sequence labeling with RNNs and transformers

# HMMs review

# HMM review

With a partner, review:

1. What are the 2 key assumptions that HMMs make?
2. What are the 2 key tables of probabilities in HMMs and what do they mean?

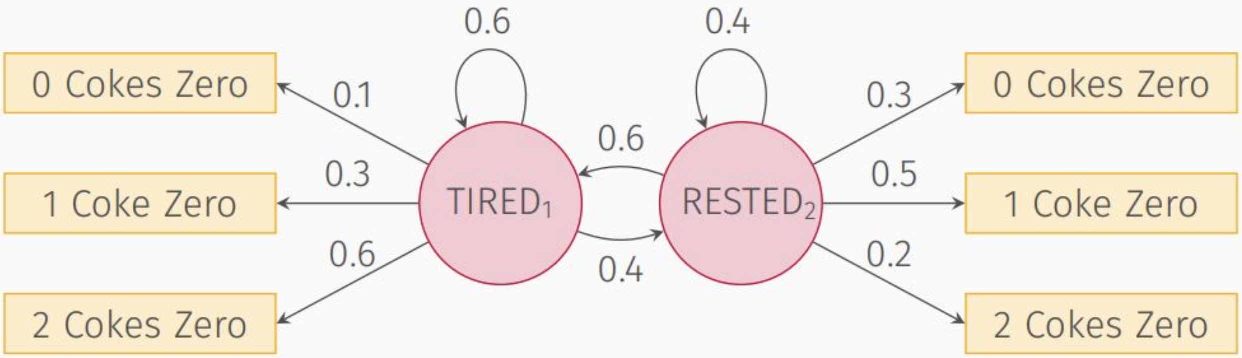
# A formal definition of the Hidden Markov Model (HMM)

| | |
|---|---|
| $Q = q_1, \ldots, q_N$ | 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). $\sum_{i=1}^{N} \pi_i = 1 \; \forall i$ |

*Slide credit: David Mortensen*

# 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?

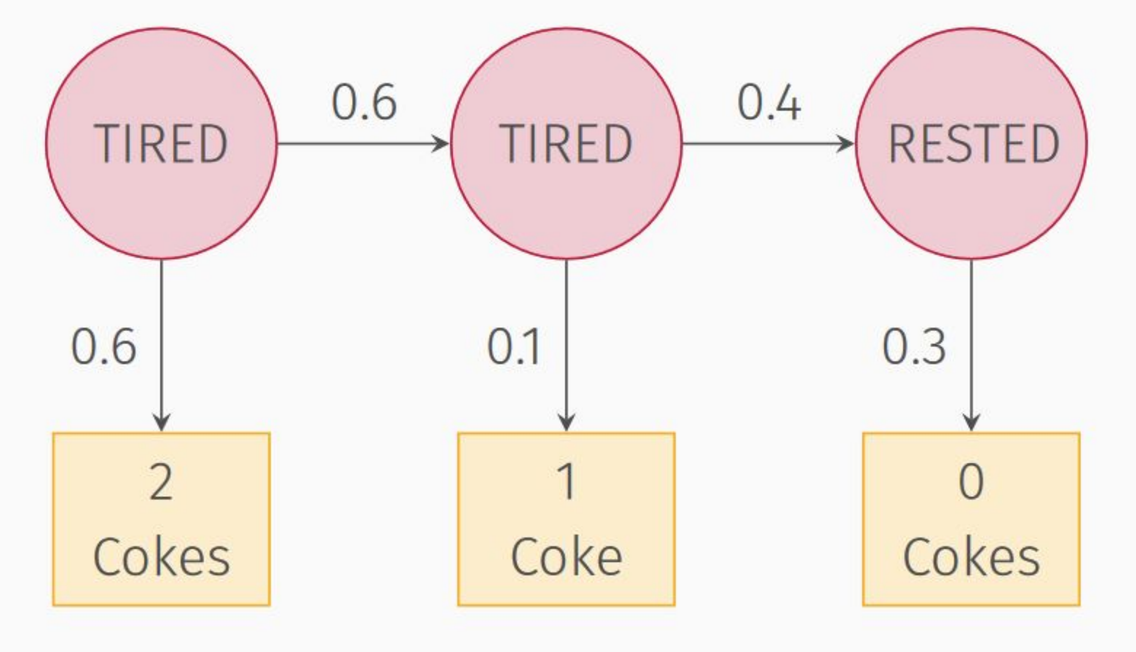



$$\pi = [0.7, 0.3]$$



*Slide credit: David Mortensen*

# An example HMM sequence






Slide adapted from David Mortensen

# Training HMMs

# Training an HMM

How do we learn the transition and emission probabilities?

- If we have (enough) data labeled with hidden and observed events, can just **use MLE/relative frequencies** with or without smoothing
- If we don't have (enough) labeled data, can use the **Forward-Backward Algorithm**, a special case of the Expectation Maximization (EM) algorithm
  - We won't go into the details of this algorithm, but the overview is that you start with an initial estimate and use that estimate to compute a better one iteratively

# Training HMMs with labeled data

Suppose we knew both the sequence of days in which a grad student is tired or rested and the number of Cokes Zero that she consumes each day:

| | | |
|---|---|---|
| 0<br>rested | 3<br>tired | 1<br>rested |
| 1<br>tired | 2<br>tired | 2<br>tired |
| 0<br>rested | 0<br>rested | 2<br>rested |

How would you train an HMM?



*Slide adapted from David Mortensen*

# Using MLE to train HMMs

First, compute $\pi$ from the initial states:

$$\pi_t = 1/3 \; \pi_r = 2/3$$

The we can compute the matrix $A$:

$$\begin{array}{ll} p(tired|tired) = 1/2 & p(tired|rested) = 1/6 \\ p(rested|tired) = 1/3 & p(rested|rested) = 2/3 \end{array}$$

and then the matrix $B$:

$$\begin{array}{ll} p(0|tired) = 0 & p(0|rested) = 2/5 \\ p(1|tired) = 1/4 & p(1|rested) = 1/5 \\ p(2|tired) = 1/2 & p(2|rested) = 1/5 \end{array}$$

*Slide adapted from David Mortensen*

# Parameters of an HMM for POS

$A =$

| | N | V | O |
|---|---|---|---|
| N | 0.1 | 0.6 | 0.3 |
| V | 0.3 | 0.3 | 0.4 |
| O | 0.3 | 0.4 | 0.3 |

transition probabilities

emission probabilities

$B =$

| | I | m | gonna | make | him | an | offer | he | can | t | refuse |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N | 0.1 | 0.00001 | 0.00001 | 0.2 | 0.1 | 0.00001 | 0.2 | 0.1 | 0.1 | 0.00001 | 0.19996 |
| V | 0.00001 | 0.1 | 0.2 | 0.2 | 0.00001 | 0.00001 | 0.05 | 0.00001 | 0.19995 | 0.00001 | 0.25 |
| O | 0.00001 | 0.00001 | 0.00001 | 0.00001 | 0.00001 | 0.5 | 0.00001 | 0.00001 | 0.00001 | 0.49991 | 0.00001 |



*Slide adapted from David Mortensen*

# Decoding HMMs: Viterbi algorithm

# 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

More formally, given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, \ldots, o_T$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_T$

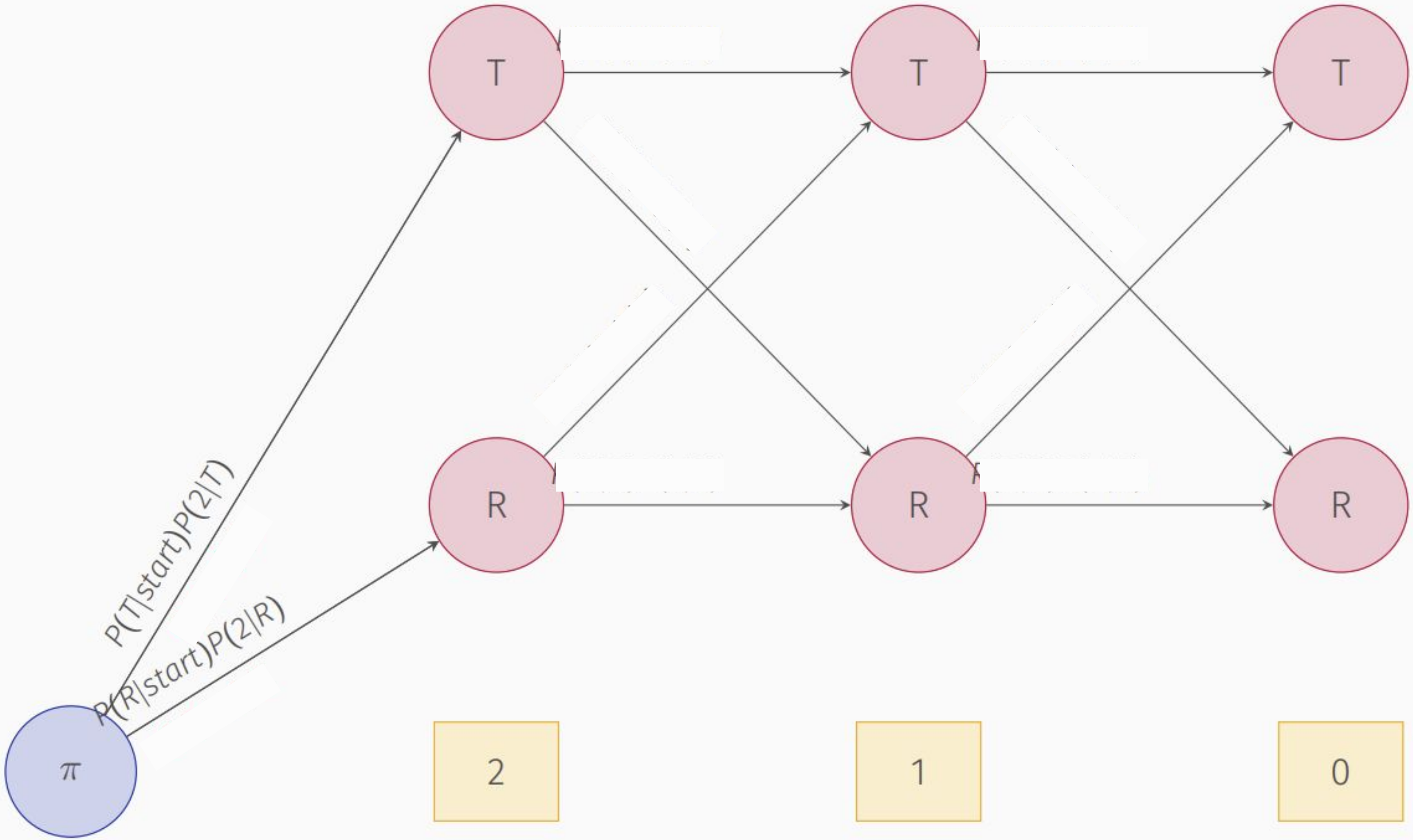


*Slide adapted from David Mortensen*

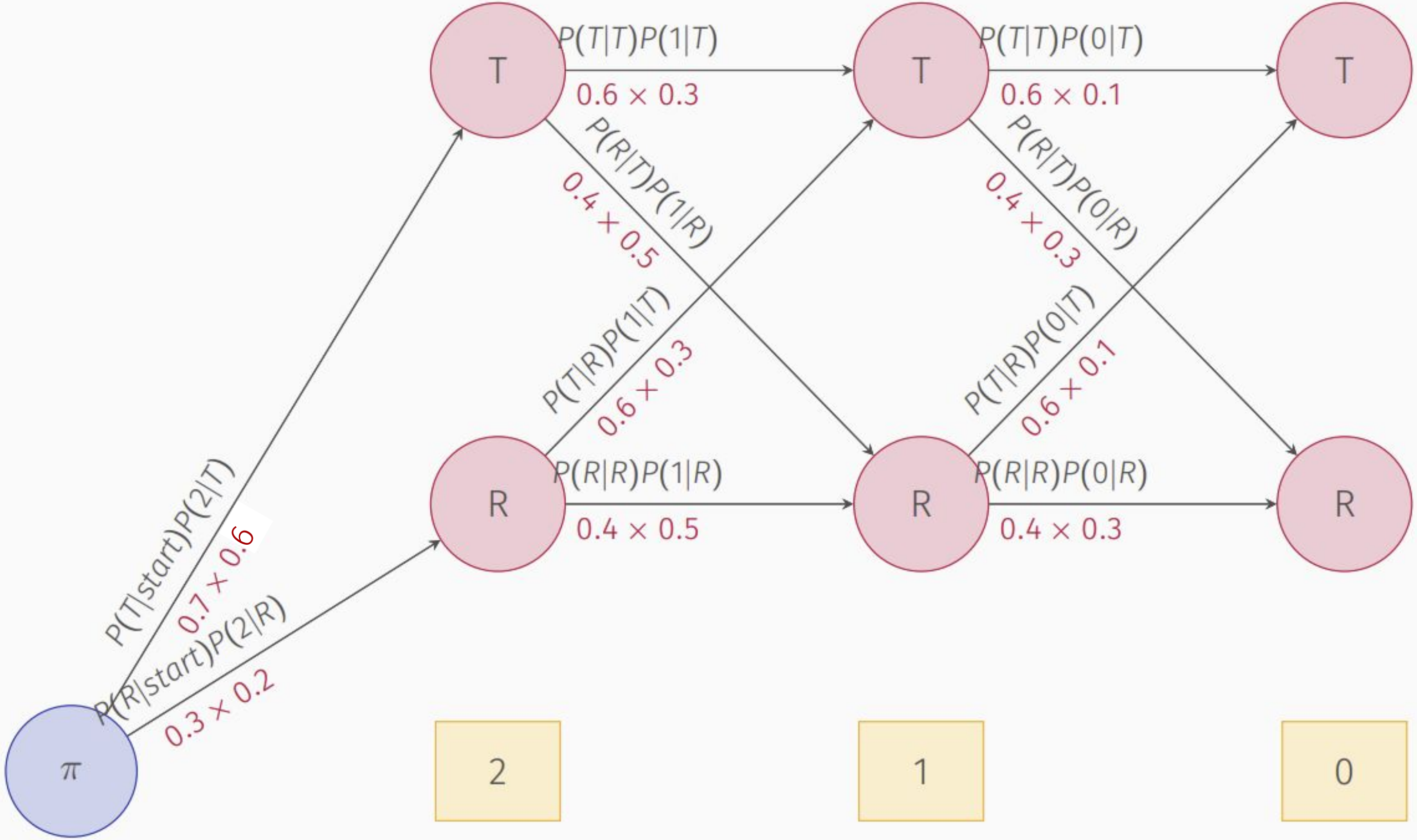
# Dynamic programming

- Solves a larger problem by combining solutions to smaller subproblems
- Fills in a table for those subproblems
- Often used in NLP to compute optimal paths through sequences

# Computing a Forward Trellis



# Computing a Forward Trellis




Slide credit: David Mortensen

# Can we do better than the Forward Algorithm for decoding?

- Computing the probability for all possible sequences of states with the forward trellis is computationally infeasible
- The set of possible state sequences (e.g. TTT, TRT, TRR, RRR, ...) grows exponentially as the number of states N grows!

**That's where dynamic programming comes in!**

- Skip the repeated computation by recording the best probabilities for subsequences along the way
- Viterbi algorithm



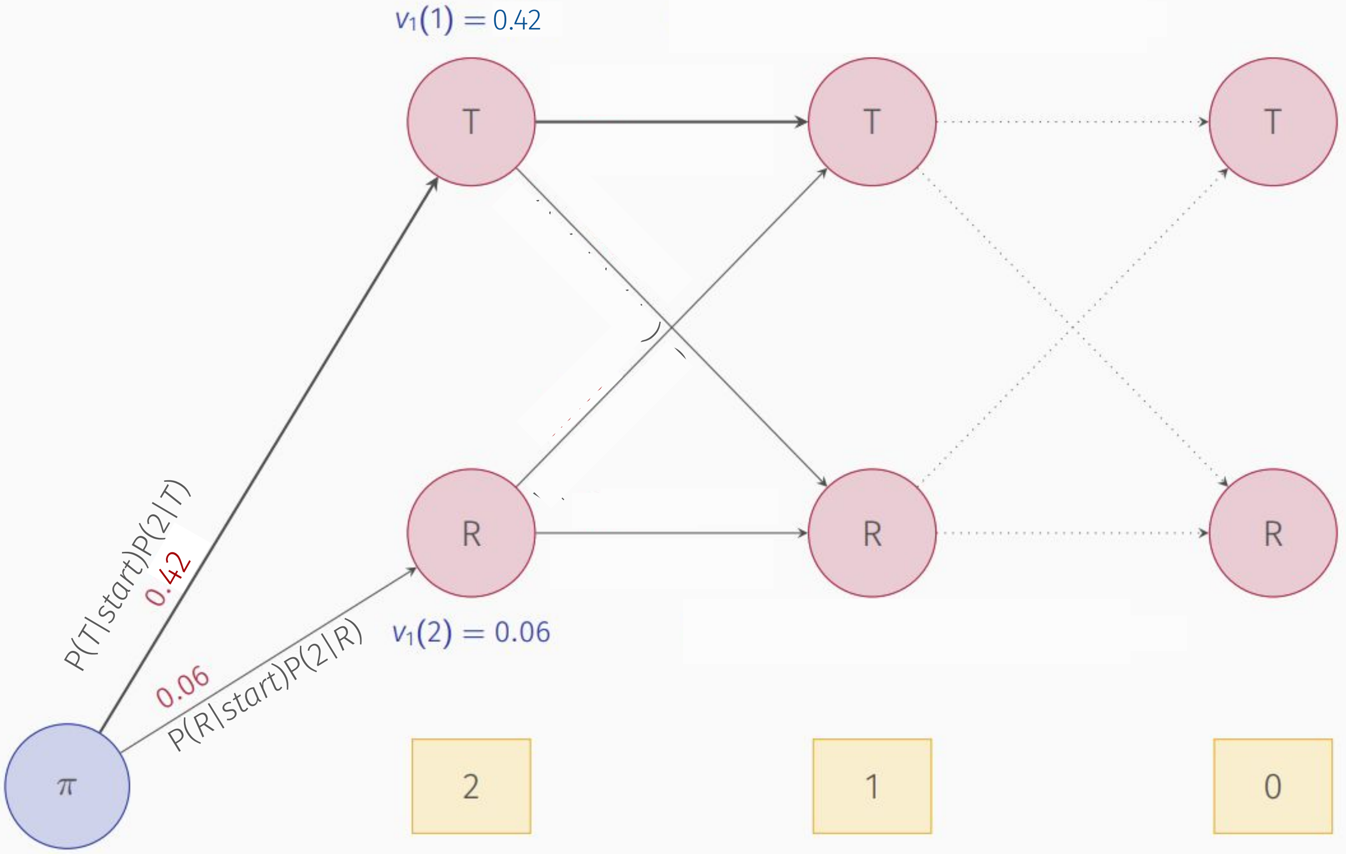
*Slide credit: David Mortensen*

# The Viterbi Algorithm Can Be Used to Decode HMMs

**function** 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**
            $V[s, t] \leftarrow \mathbf{max}_{s'=1}^{N} V[s', t-1] \cdot a_{s',s} \cdot b_s(o_t)$
            $B[s, t] \leftarrow \mathbf{argmax}_{s'=1}^{N} V[s', t-1] \cdot a_{s',s} \cdot b_s(o_t)$
    *bestpathprob* $\leftarrow \mathbf{max}_{s=1}^{N} V[s, T]$
    *bestpathpointer* $\leftarrow \mathbf{max}_{s=1}^{N} V[s, T]$
    *bestpath* $\leftarrow$ path starting at *bestpathpointer* that follows $b$ to states back in time.
    **return** *bestpath*, *bestpathprob*

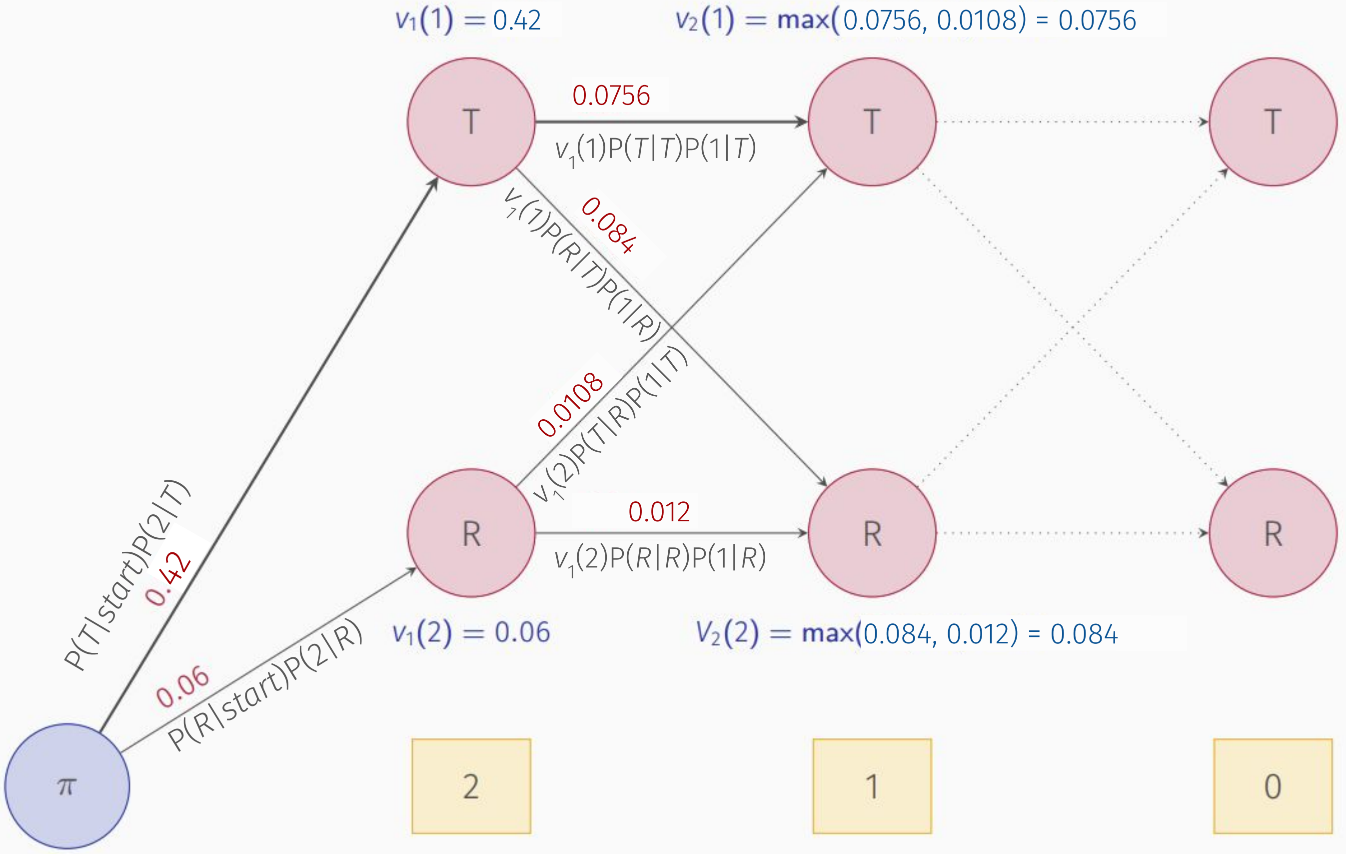
*Slide credit: David Mortensen*

# Using Viterbi to Decode an HMM




*Slide adapted from David Mortensen*

# Using Viterbi to Decode an HMM





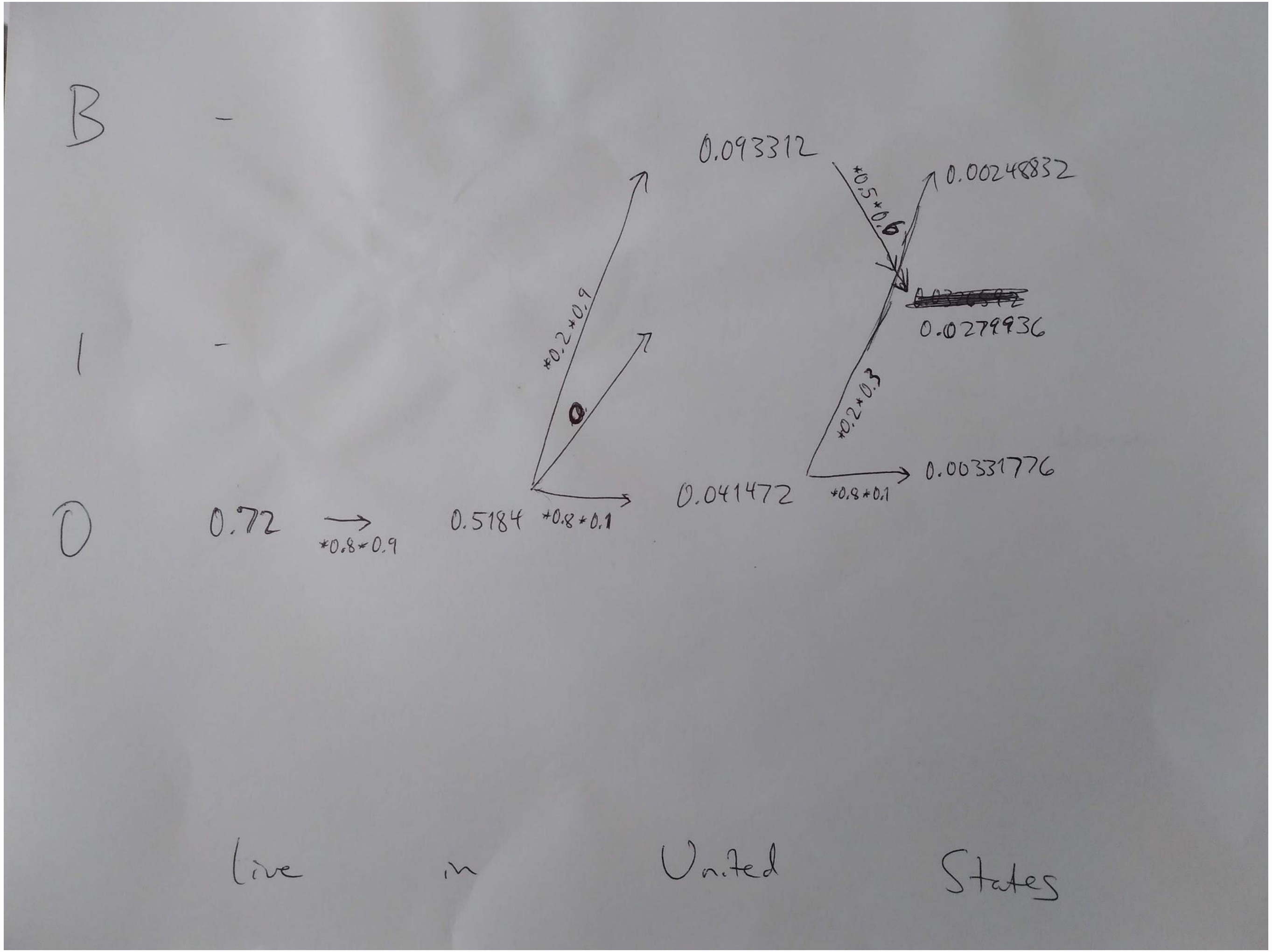

*Slide adapted from David Mortensen*

| | B | I | O |
|---|---|---|---|
| B | 0 | 0.5 | 0.5 |
| I | .1 | 0 | 0.9 |
| O | 0.2 | 0 | 0.8 |

| | United | States | live | in |
|---|---|---|---|---|
| B | 0.8 | 0.3 | 0 | 0 |
| I | 0.1 | 0.6 | 0.1 | 0.1 |
| O | 0.1 | 0.1 | 0.9 | 0.9 |

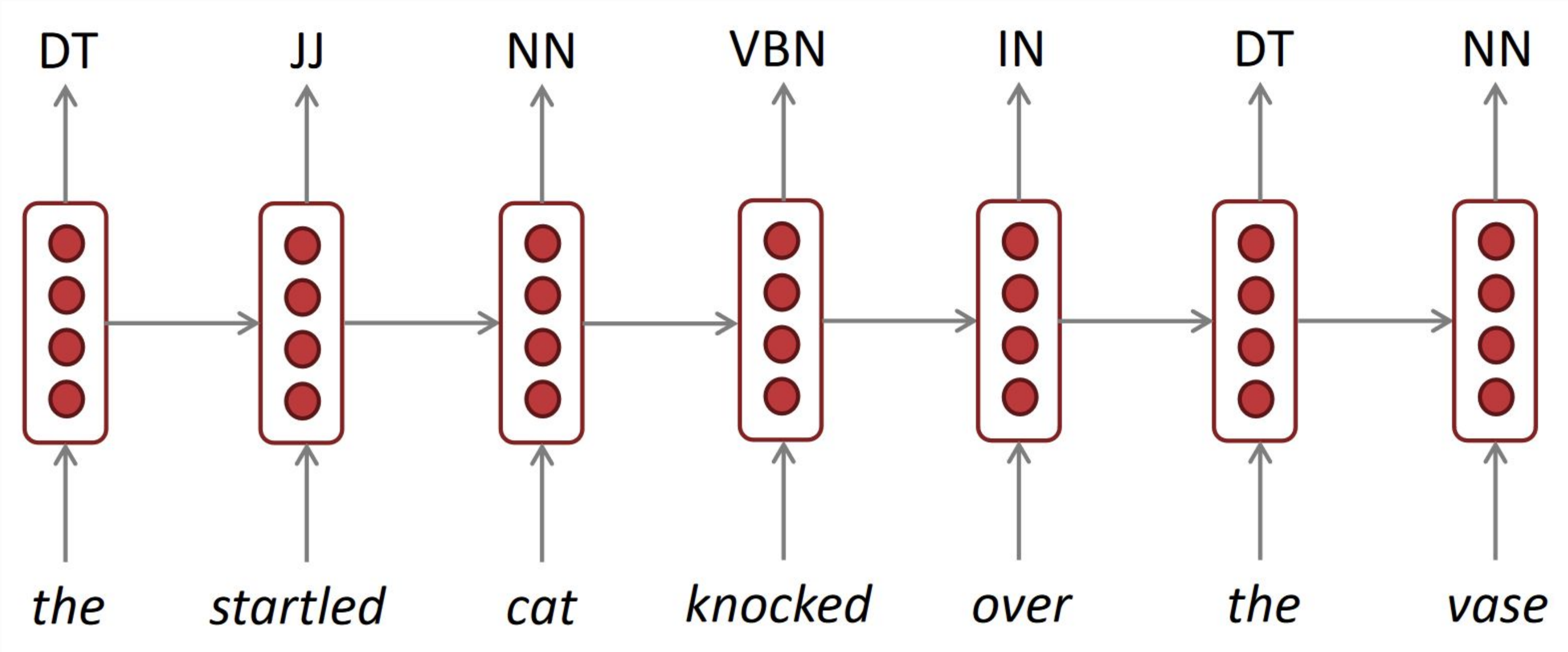
| | $\pi$ |
|---|---|
| B | 0.2 |
| I | 0 |
| O | 0.8 |

To decode:
live in United States


B
I
O
0.72
*0.8*0.9
0.5184
*0.8*0.1
*0.2*0.9
0
0.093312
*0.5*0.6
0.00248832
0.0279936
*0.2*0.3
0.041472
*0.8*0.1
0.00331776
live
in
United
States

# Neural sequence labeling

# RNNs can be used for sequence labeling






*Slide adapted from Chris Manning*

# BERT can be used for sequence labeling

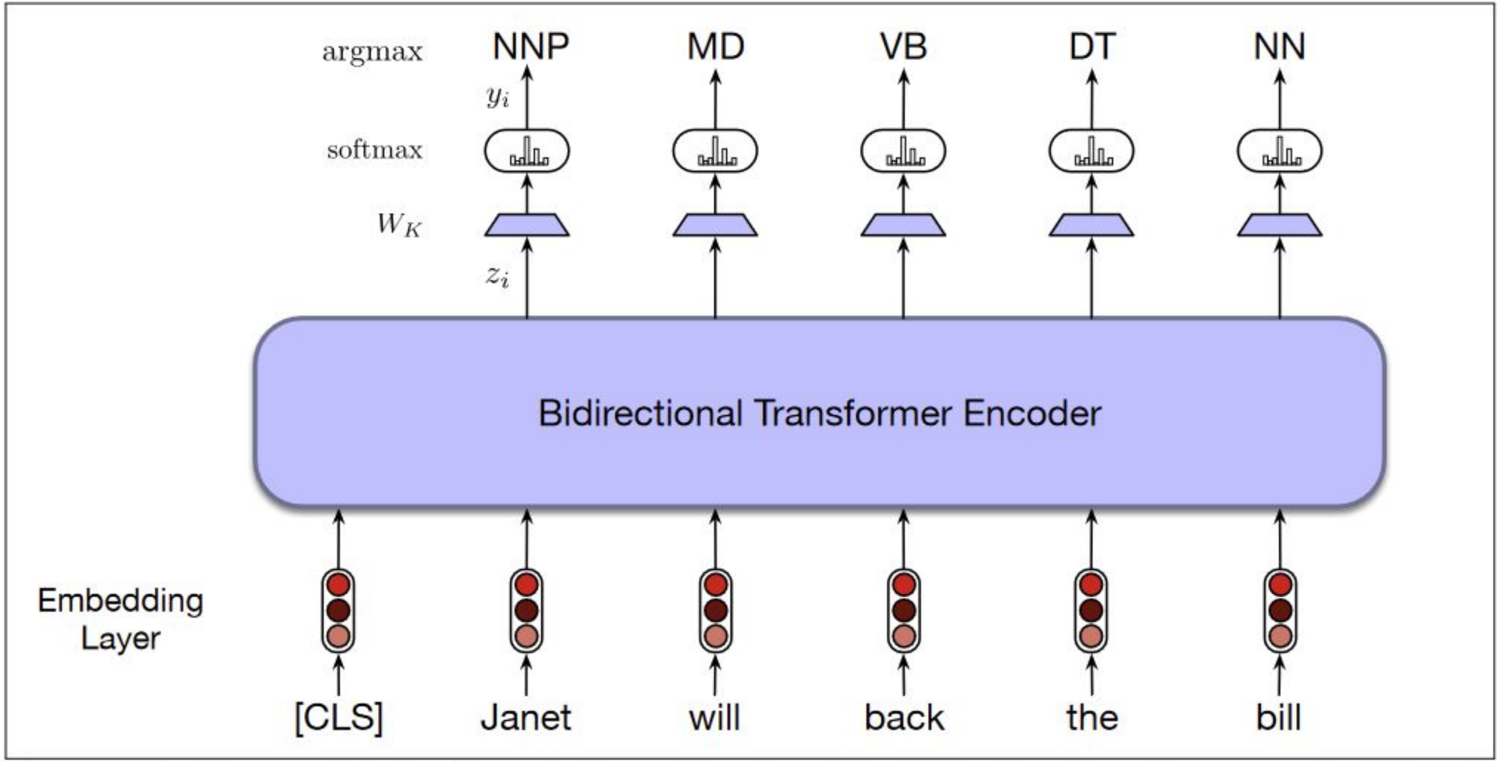



**Figure 11.9** Sequence labeling for part-of-speech tagging with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

*Slide adapted from Jurafsky & Martin*

# An alternative to BIO: span-based NER

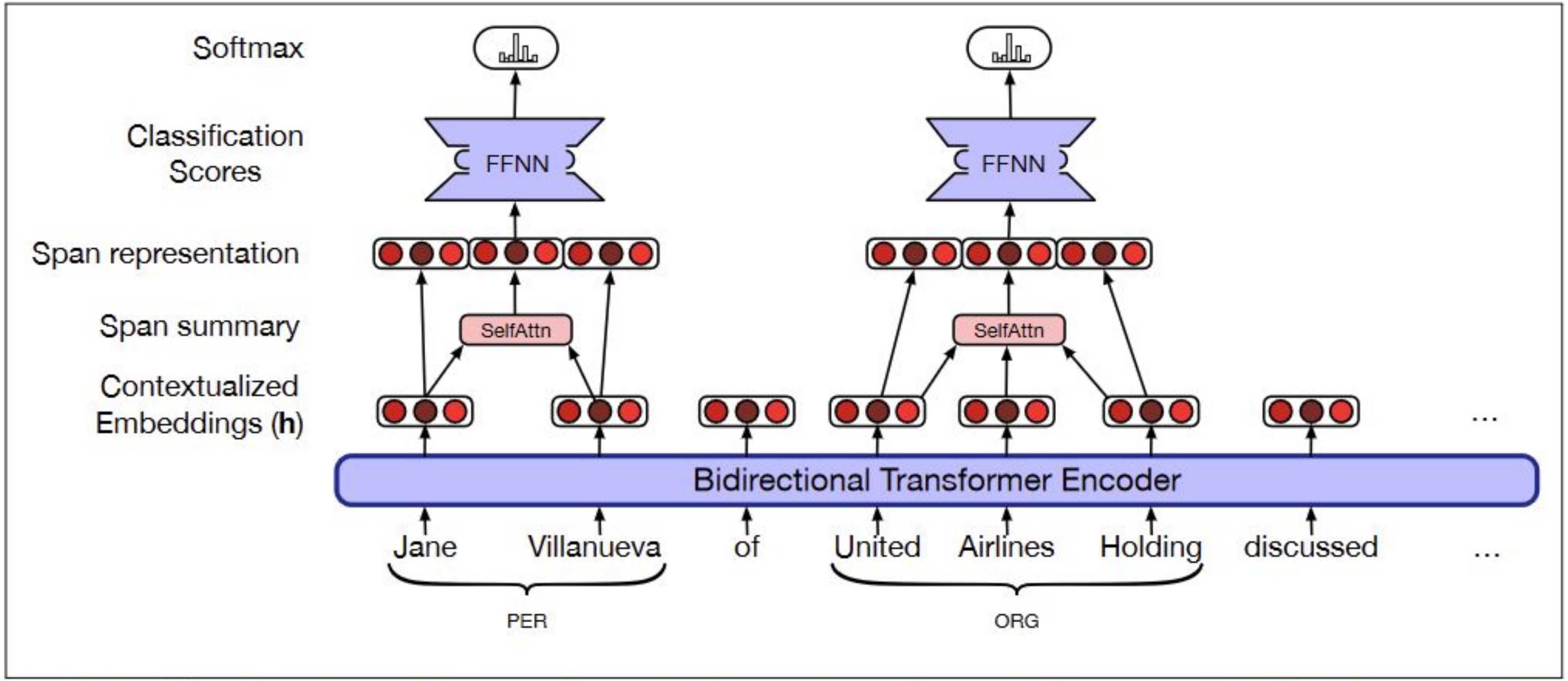



**Figure 11.10** A span-oriented approach to named entity classification. The figure only illustrates the computation for 2 spans corresponding to ground truth named entities. In reality, the network scores all of the $\frac{T(T-1)}{2}$ spans in the text. That is, all the unigrams, bigrams, trigrams, etc. up to the length limit.

*Slide adapted from Jurafsky & Martin*

# Wrapping up

- If enough annotated training data is available, HMMs can be trained with MLE
- The Viterbi algorithm is used for decoding HMMs
- RNNs and transformers can be trained to do sequence labeling

# *Questions?*