ILLINOIS INSTITUTE OF TECHNOLOGY

Hidden Markov models and the Viterbi algorithm

CS-585

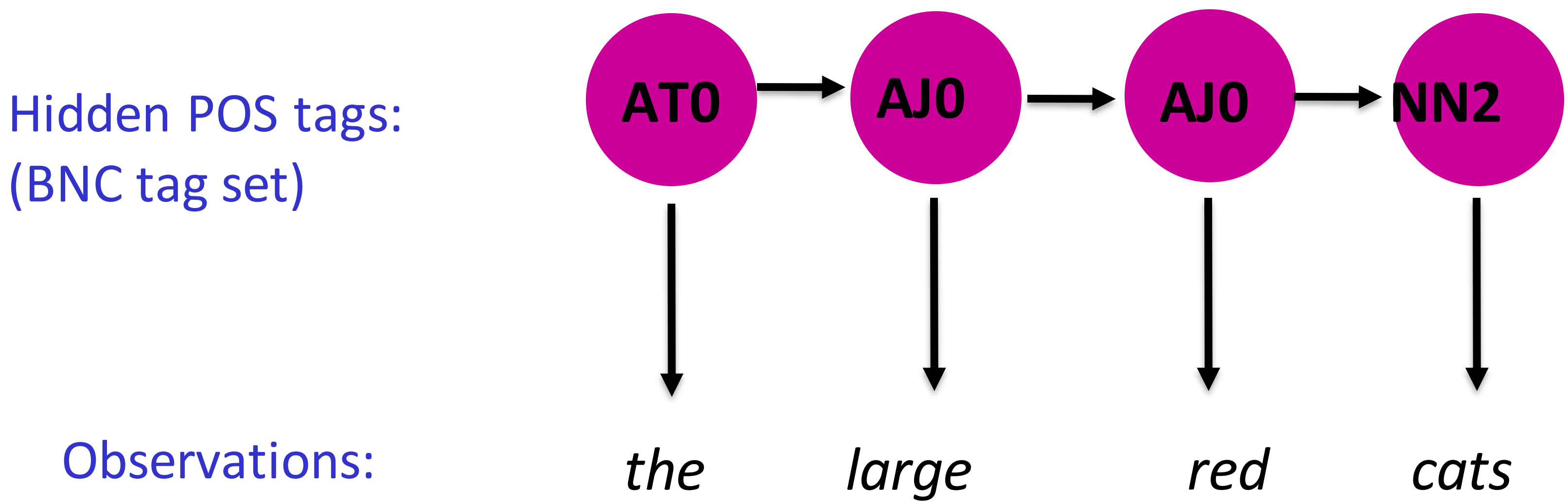
Natural Language Processing

Sonjia Waxmonsky

Hidden Markov Model (HMM)

- A *generative* framework for sequence labeling
  - Expresses a joint probability distribution $P(t_{1..n}, w_{1..n})$ over the observed word sequence and unobserved tag/label sequence
  - A "generative story" according to which each word is generated according to a distribution dependent on a fixed-length tag history

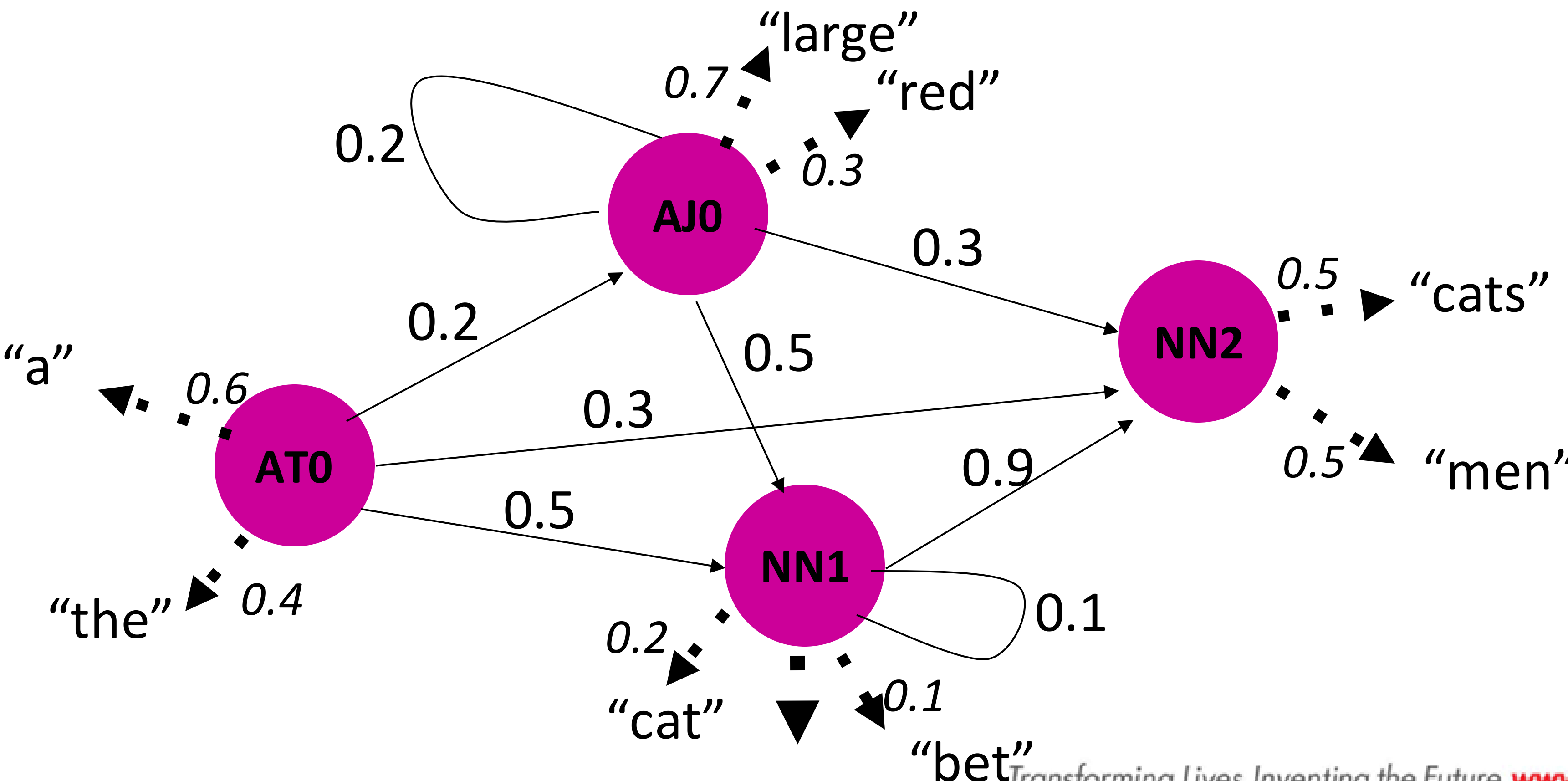
Hidden Markov Model (HMM)



Based on Figure 7.2 (E-NLP p.155)

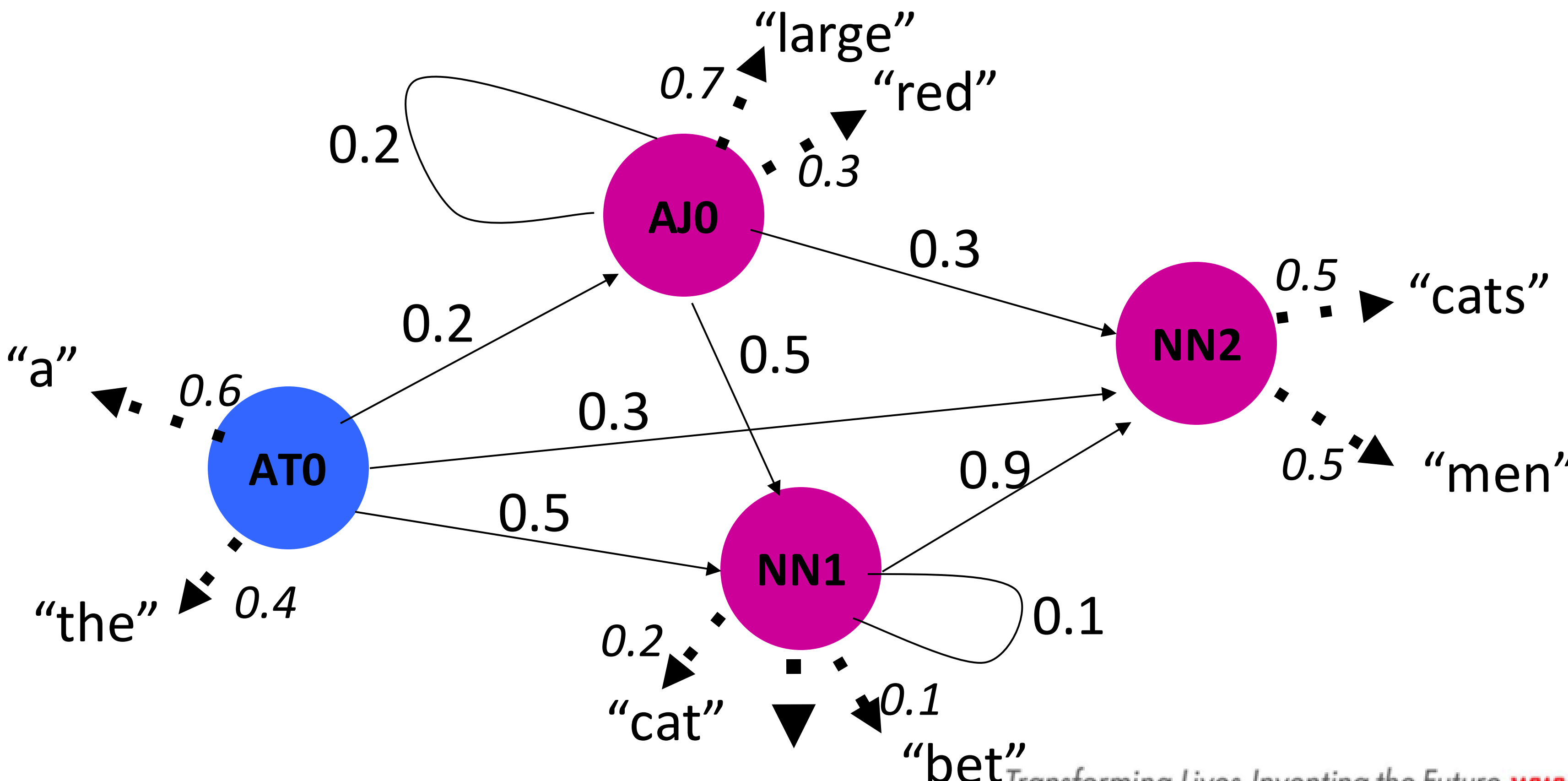
Hidden Markov Model (HMM)

- **Assumption:** POS generated as random process, and each POS randomly generates a word



Hidden Markov Model (HMM)

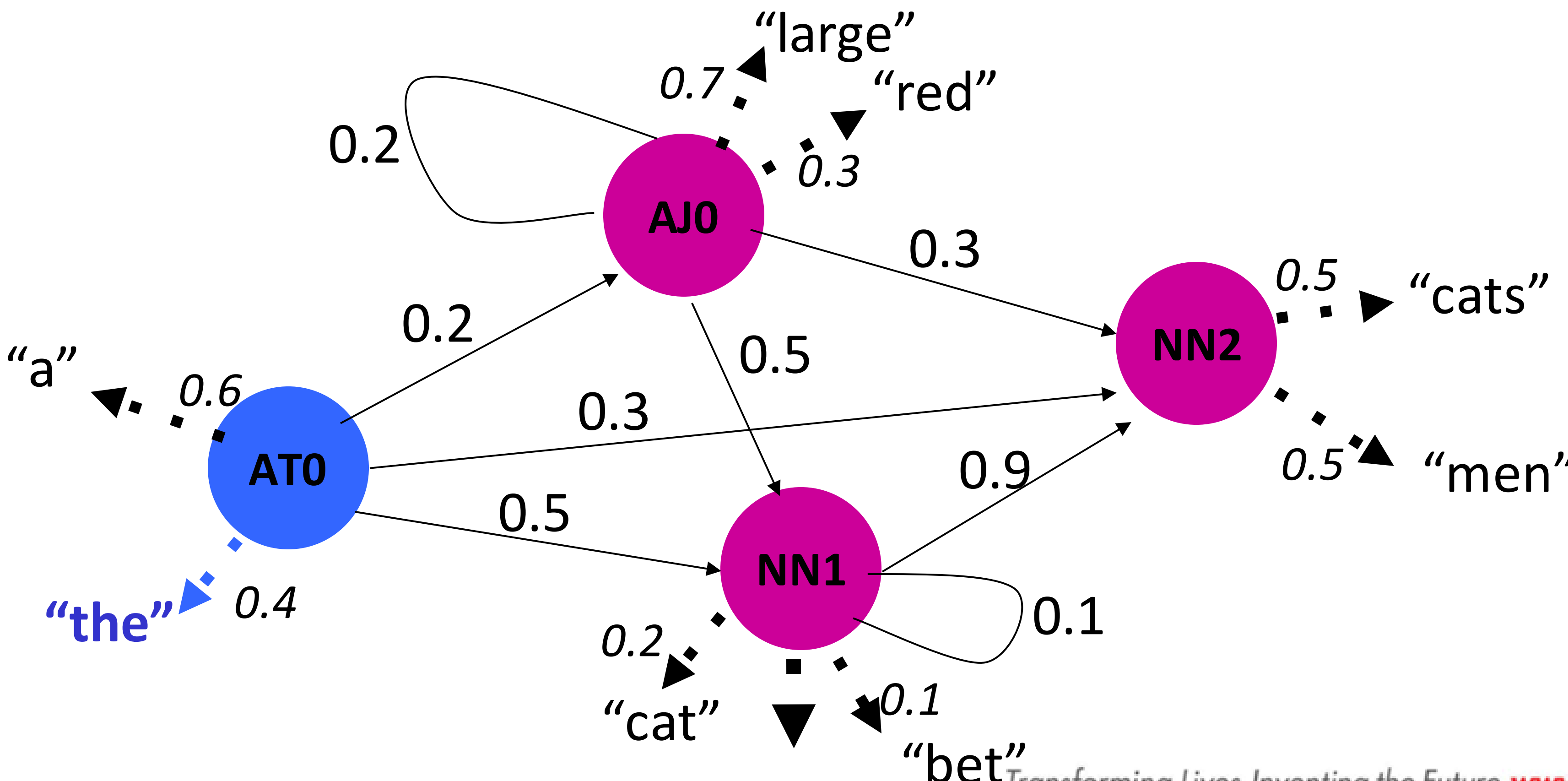
AT0



Hidden Markov Model (HMM)

the

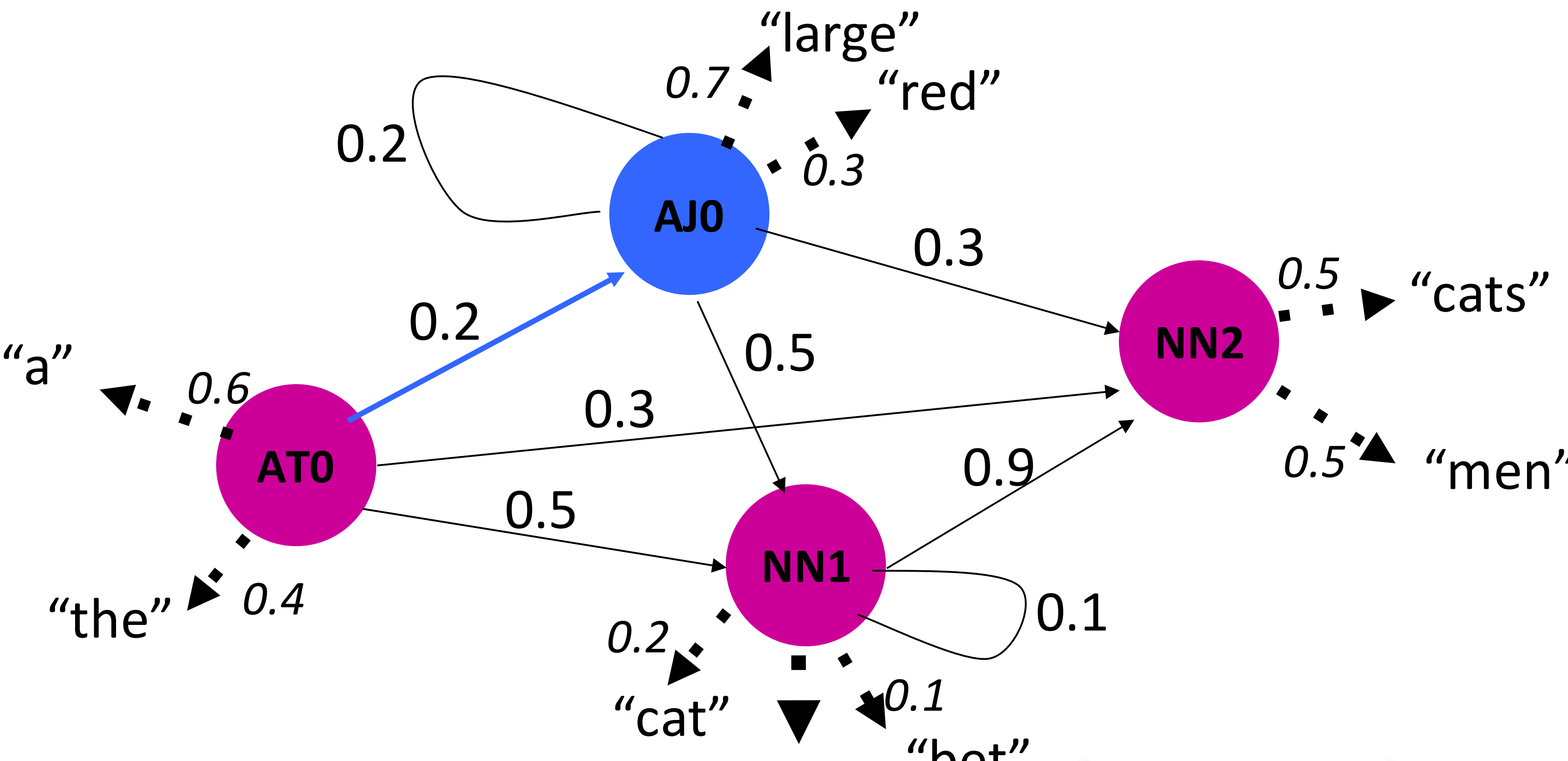
AT0



Hidden Markov Model (HMM)

the

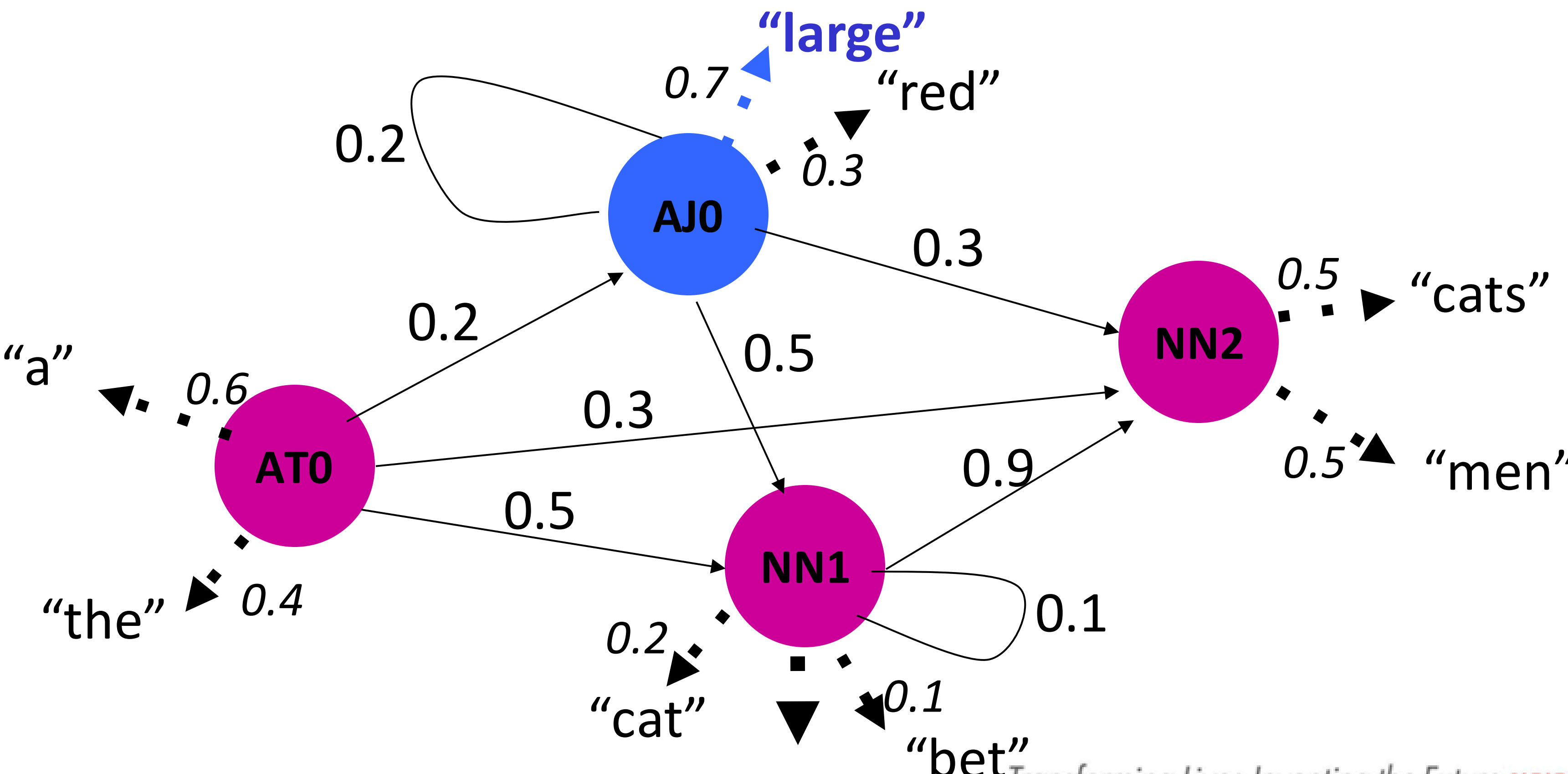
AT0   **AJ0**



Hidden Markov Model (HMM)

the *large*

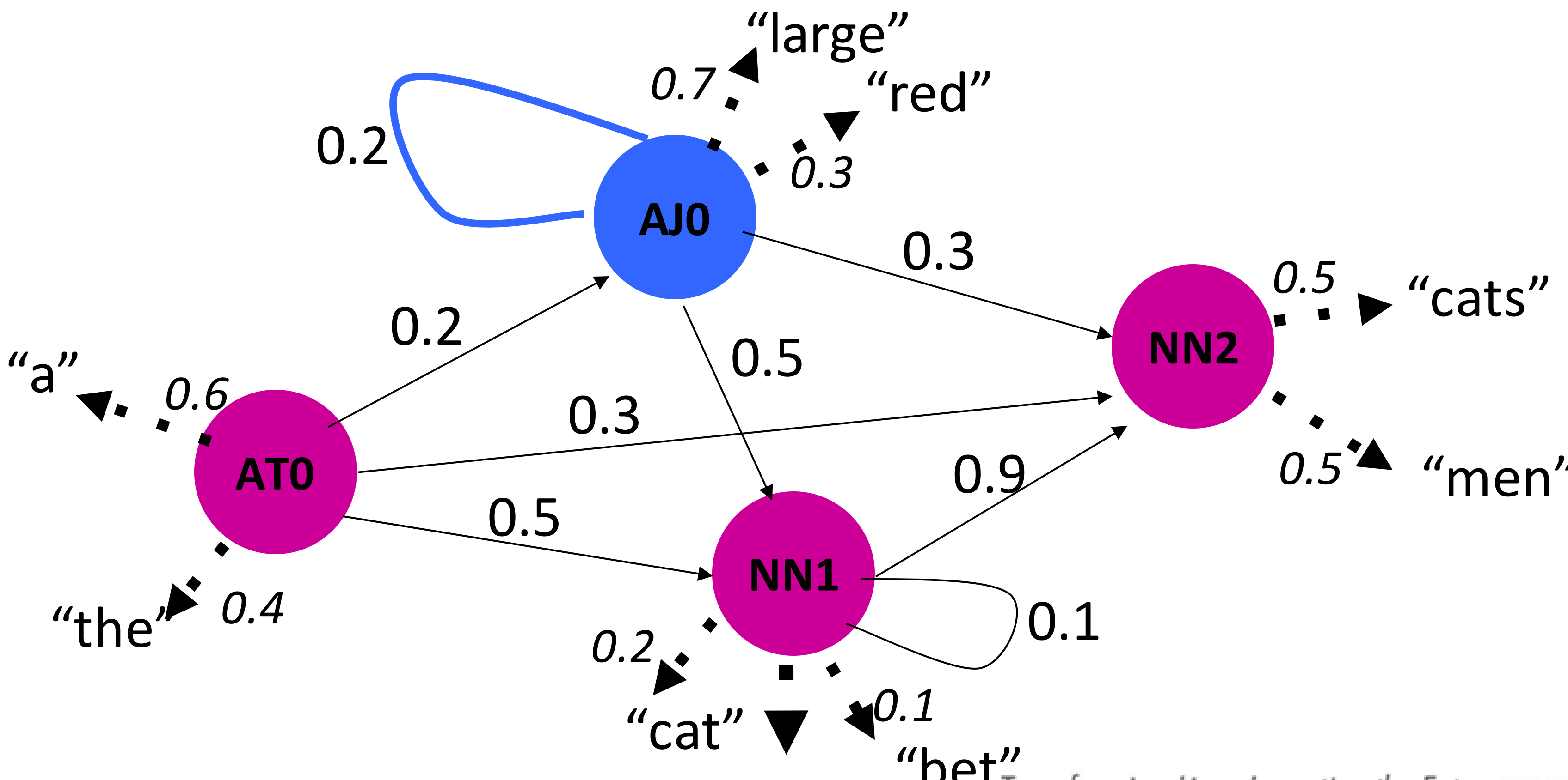
AT0 **AJ0**



Hidden Markov Model (HMM)

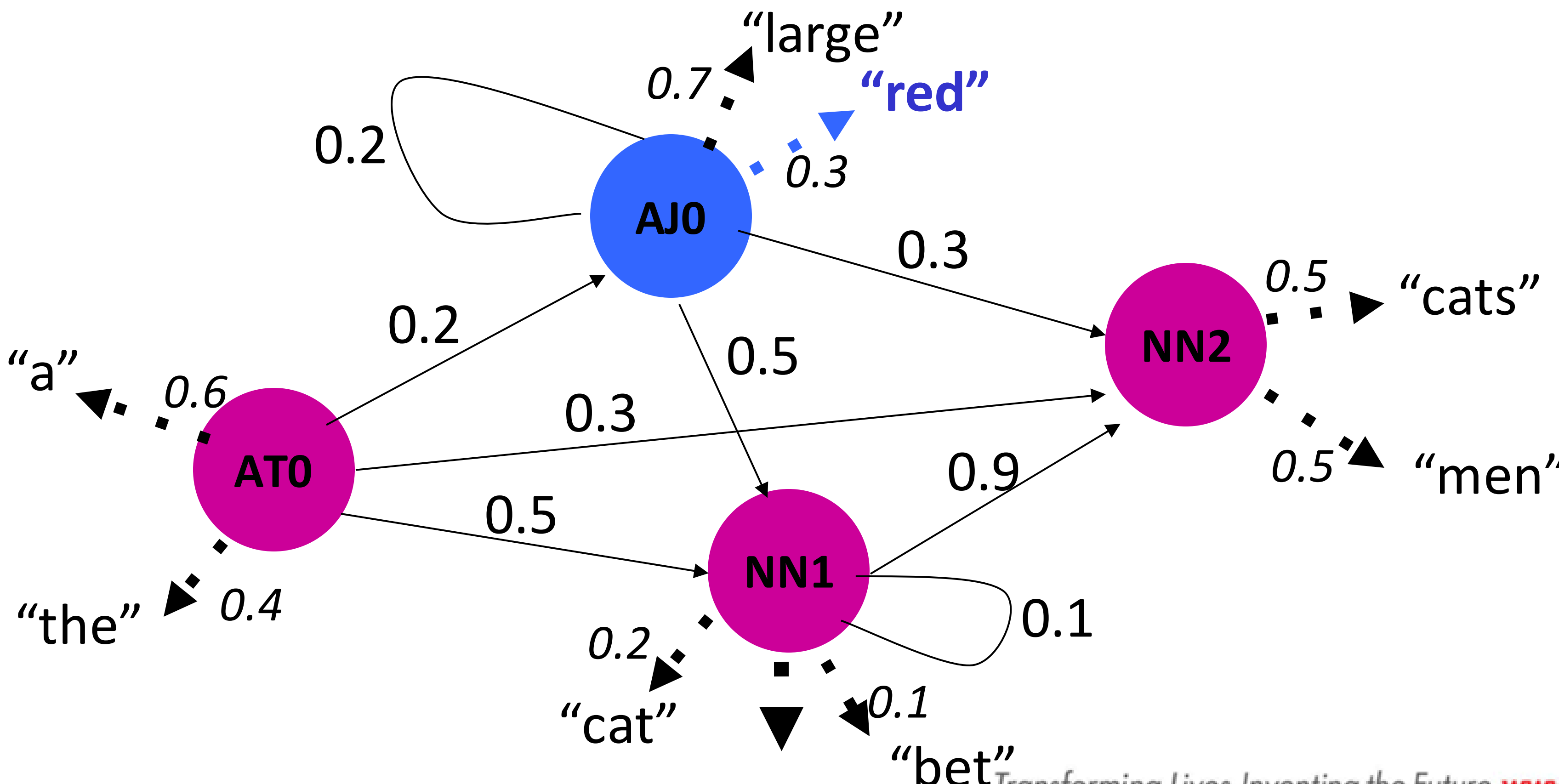
the *large*

AT0 AJ0 AJ0



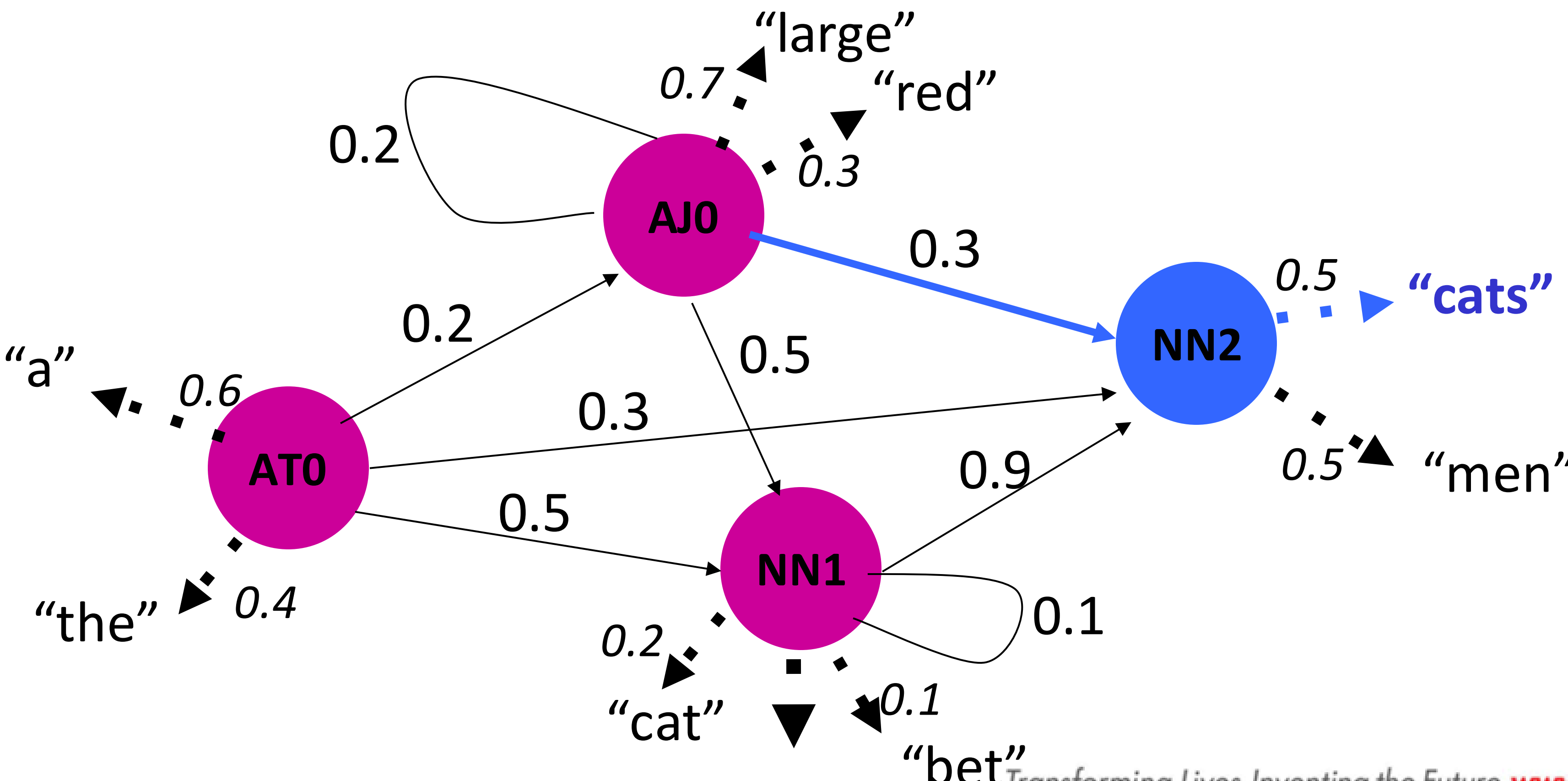
Hidden Markov Model (HMM)

the	*large*	*red*
AT0	**AJ0**	**AJ0**



Hidden Markov Model (HMM)

the	*large*	*red*	*cats*
AT0	**AJ0**	**AJ0**	**NN2**



HMM – POS generation

- First-order (bigram) Markov assumptions:
  - Limited Horizon: Tag depends only on previous tag

$$P\left(t_{i+1} = t^k | t_1 = t^{j_1}, \dots, t_i = t^{j_i}\right) = P(t_{i+1} = t^k | t_i = t^{j_i})$$

  - Time invariance: No change over time

$$P\left(t_{i+1} = t^k | t_i = t^j\right) = P\left(t_2 = t^k \middle| t_1 = t^j\right) = P(t^j \to t^k)$$

HMM – Word generation

- Output probabilities:
  - Probability of getting word w^k for tag t^j:

$$P(w^k|t^j)$$

Assumption:

Not dependent on other tags or words!

Combining Probabilities

Probability of a tag sequence:

$$P(t_1, t_2, \ldots, t_N) = P(t_1)P(t_1 \to t_2) \ldots P(t_{N-1} \to t_N)$$

Assume t_0 = "universal" start tag:

$$= P(t_0 \to t_1)P(t_1 \to t_2) \ldots P(t_{N-1} \to t_N)$$

$$= \prod_i P(t_{i-1} \to t_i)$$

Prob. of word sequence *and* tag sequence:

$$P(W, T) = \prod_i P(t_{i-1} \to t_i)\, P(w_i | t_i)$$

Training from labeled data

- Labeled training = each word has a POS tag
- Thus:

$$P_{MLE}(t^j) = \frac{C(t^j)}{N}$$

$$P_{MLE}(t^j \rightarrow t^k) = \frac{C(t^j, t^k)}{C(t^j)}$$

$$P_{MLE}(w^k | t^j) = \frac{C(t^j : w^k)}{C(t^j)}$$

$$P_{MLE}(t^j | w^k) = \frac{C(t^j : w^k)}{C(w^k)}$$

Three Basic POS Computations

Model *m* contains transition and output probabilities

- Compute the probability of a text:

$$\boldsymbol{P_m(W_{1,N})}$$

- Compute maximum probability tag sequence:

$$\operatorname*{argmax}_{T_{1,N}} P_m(T_{1,N} | W_{1,N})$$

- Compute maximum likelihood model

$$\operatorname*{argmax}_{m} P_m(W_{1,N})$$

$P_m(W_{1,N})$: Forward Algorithm

Define $a_k(i) = P(w_{1,k}, t_k = t^i)$

for i in $[1, \ldots, N_t]$:

$a_1(i) \leftarrow P_m(t_0 \rightarrow t^i) P_m(w_1 | t^i)$

for k in $[2, \ldots, N]$ ← Loop over words

for j in $[1, \ldots, N_t]$: ← Loop over tags

$a_k(j) \leftarrow \left(\sum_i a_{k-1}(i) P_m(t^i \rightarrow t^j) \right) P_m(w_k | t^j)$

$P_m(W_{1,N}) = \sum_i a_N(i)$ ← Sum over tags

Complexity $= O(N_t^{\,2} N)$ ← N_t = # tags, N=length(sequence)

$P_m(W_{1,N})$: Forward Algorithm

Define $a_k(i) = P(w_{1,k}, t_k = t^i)$

for i in $[1, \ldots, N_t]$ **:**

$a_1(i) \leftarrow P_m(t_0 \rightarrow t^i) P_m(w_1 | t^i)$

Initialize: probability of generating the first word and tag

for k in $[2, \ldots, N]$

for j in $[1, \ldots, N_t]$ **:**

$a_k(j) \leftarrow \left(\sum_i a_{k-1}(i) P_m(t^i \rightarrow t^j) \right) P_m(w_k | t^j)$

$P_m(W_{1,N}) = \sum_i a_N(i)$

Complexity $= O(N_t^{\,2} N)$

$P_m(W_{1,N})$: Forward Algorithm

Define $a_k(i) = P(w_{1,k}, t_k = t^i)$

For each index k,
For each tag j,
Sum probabilities across prior tags that could be transitioned from

for i in $[1, \ldots, N_t]$ **:**

$a_1(i) \leftarrow P_m(t_0 \rightarrow t^i) P_m(w_1 | t^i)$

for k in $[2, \ldots, N]$

for j in $[1, \ldots, N_t]$ **:**

$a_k(j) \leftarrow \left(\sum_i a_{k-1}(i) P_m(t^i \rightarrow t^j)\right) P_m(w_k | t^j)$

$P_m(W_{1,N}) = \sum_i a_N(i)$

Complexity $= O(N_t^{\,2} N)$

$P_m(W_{1,N})$: Forward Algorithm

Define $a_k(i) = P(w_{1,k}, t_k = t^i)$

for i in $[1, \ldots, N_t]$ **:**

$a_1(i) \leftarrow P_m(t_0 \rightarrow t^i) P_m(w_1 | t^i)$

for k in $[2, \ldots, N]$

for j in $[1, \ldots, N_t]$ **:**

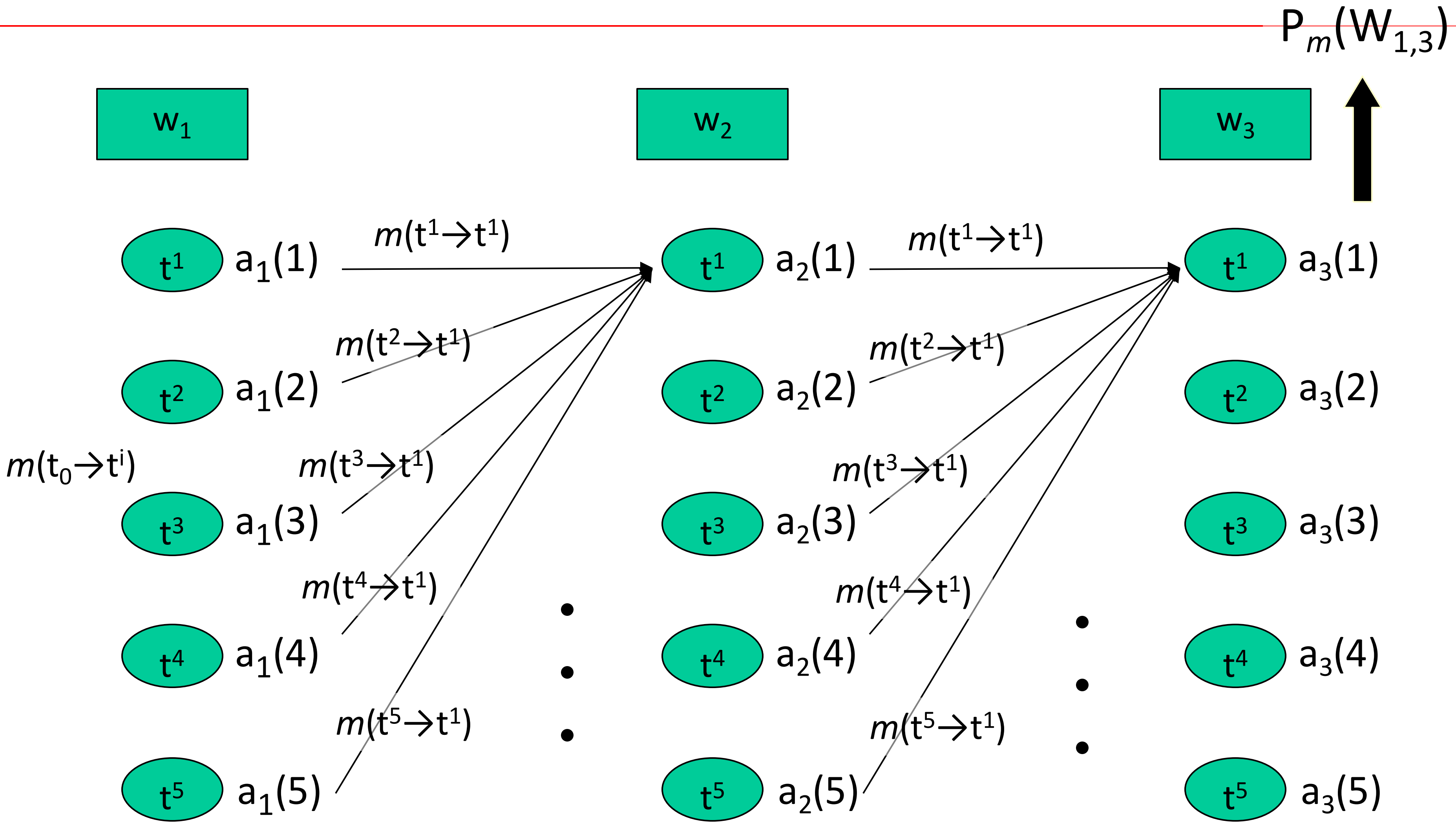
$a_k(j) \leftarrow \left(\sum_i a_{k-1}(i) P_m(t^i \rightarrow t^j) \right) P_m(w_k | t^j)$

$P_m(W_{1,N}) = \sum_i a_N(i)$

To get forward probability of whole sequence, loop over indices, loop over tags, and sum over tags

Complexity $= O(N_t^{\,2} N)$

Forward algorithm



$P_m(W_{1,N})$: Backward Algorithm

Define $b_k(i) = P(w_{k+1,N} | t_k = t^i)$

for i in $[1, \dots, N_t]$:

$b_N(i) \leftarrow 1$

for k in $[N-1, \dots, 1]$

for j in $[1, \dots, N_t]$:

$b_k(j) \leftarrow \sum_i P_m(t^j \rightarrow t^i)\ P_m(w_{k+1} | t^i)\ b_{k+1}(i)$

$$P_m(W_{1,N}) = \sum_i P_m(t_0 \rightarrow t^i)\ P_m(w_1 | t^i)\ b_1(i)$$

Complexity $= O(N_t^2 N)$

$P_m(W_{1,N})$: Backward Algorithm

Define $b_k(i) = P(w_{k+1,N} | t_k = t^i)$

for i in $[1, \ldots, N_t]$ **:**

$b_N(i) \leftarrow 1$

Initialize: probability of ending up at the end of the sequence

for k in $[N-1, \ldots, 1]$

for j in $[1, \ldots, N_t]$ **:**

$b_k(j) \leftarrow \sum_i P_m(t^j \rightarrow t^i)\ P_m(w_{k+1} | t^i)\ b_{k+1}(i)$

$$P_m(W_{1,N}) = \sum_i P_m(t_0 \rightarrow t^i)\ P_m(w_1 | t^i)\ b_1(i)$$

$$\text{Complexity} = O(N_t^2 N)$$

$P_m(W_{1,N})$: Backward Algorithm

Define $b_k(i) = P(w_{k+1,N} | t_k = t^i)$

For each index k,
For each tag j,
Sum probabilities across following tags that could be transitioned to

$\texttt{for}\ \ i \text{ in } [1, \dots, N_t]\ \ \texttt{:}$

$\quad b_N(i) \leftarrow 1$

$\texttt{for}\ \ k \text{ in } [N-1, \dots, 1]$

$\quad \texttt{for}\ \ j \text{ in } [1, \dots, N_t]\ \ \texttt{:}$

$\qquad b_k(j) \leftarrow \sum_i P_m(t^j \to t^i)\ P_m(w_{k+1} | t^i)\ b_{k+1}(i)$

$$P_m(W_{1,N}) = \sum_i P_m(t_0 \to t^i)\ P_m(w_1 | t^i)\ b_1(i)$$

$$\text{Complexity} = O(N_t^{\,2} N)$$

$P_m(W_{1,N})$: Backward Algorithm

Define $b_k(i) = P(w_{k+1,N} | t_k = t^i)$

for i in $[1, \dots, N_t]$:

$\quad b_N(i) \leftarrow 1$

for k in $[N-1, \dots, 1]$

$\quad$ for j in $[1, \dots, N_t]$:

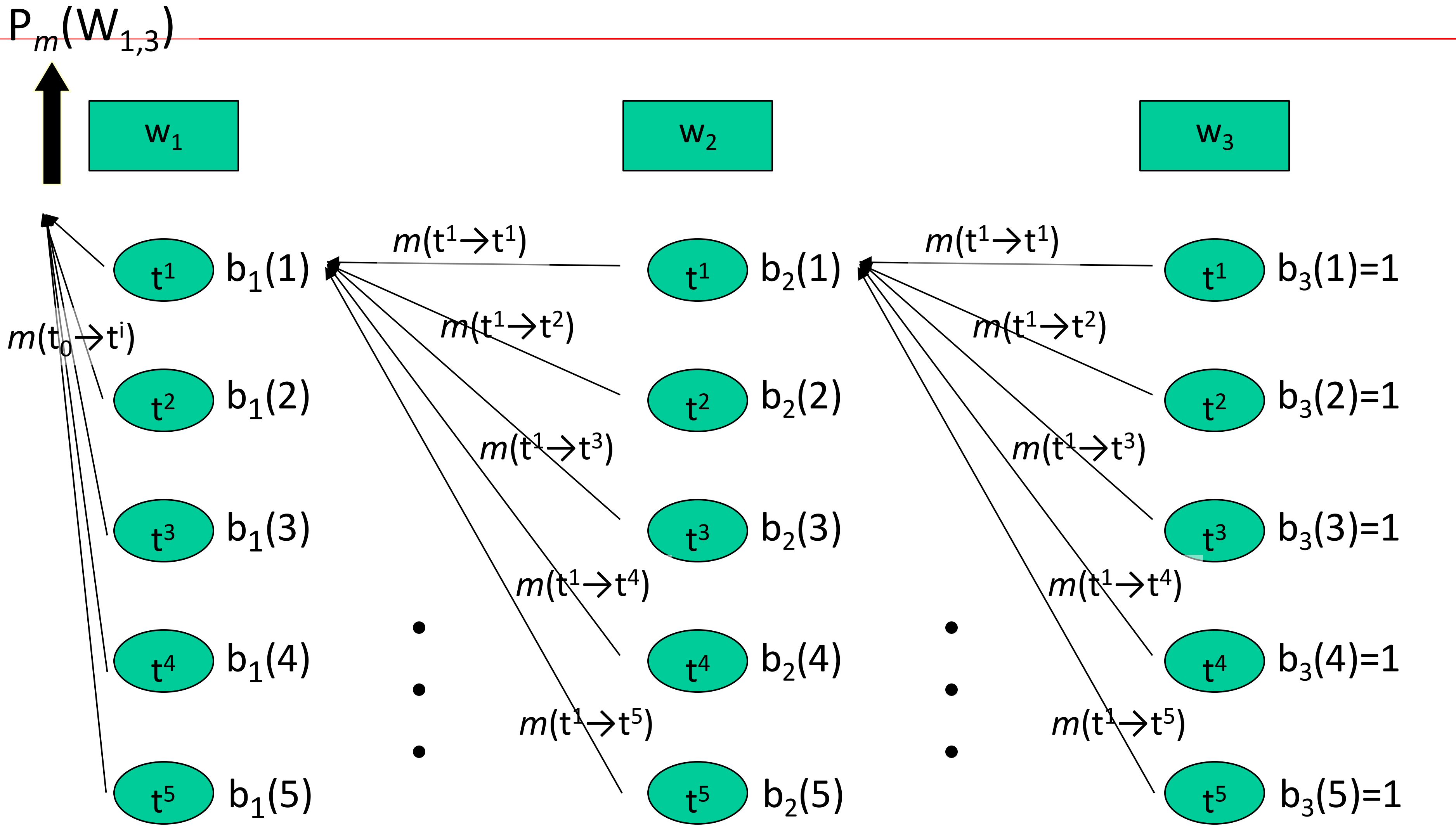
$\quad\quad b_k(j) \leftarrow \sum_i P_m(t^j \to t^i)\ P_m(w_{k+1} | t^i)\ b_{k+1}(i)$

To get backward probability of whole sequence, loop over indices, loop over tags, and sum over tags

$P_m(W_{1,N}) = \sum_i P_m(t_0 \to t^i)\ P_m(w_1 | t^i)\ b_1(i)$

Complexity $= O(N_t^{\,2} N)$

Backward algorithm



P(W)

- So, using forward probabilities:

$$P_m(W_{1,N}) = \sum_i a_N(i)$$

- Using backward probabilities:

$$P_m(W_{1,N}) = \sum_i P_m(t_0 \to t^i)\ P_m(w_1|t^i)\ b_1(i)$$

- Using both:

$$P_m(W_{1,N}) = \sum_i a_r(i)\, b_r(i)$$

Three Basic POS Computations

Model *m* contains transition and output probabilities

- Compute the probability of a text:

$$P_m(W_{1,N})$$

- Compute maximum probability tag sequence:

$$\underset{\boldsymbol{T_{1,N}}}{\textbf{argmax}}\, \boldsymbol{P_m(T_{1,N}|W_{1,N})}$$

- Compute maximum likelihood model

$$\underset{m}{\operatorname{argmax}}\, P_m(W_{1,N})$$

Viterbi Tagging

Most probable tag sequence given text:

$$T^* = \text{argmax}_T \, P_m(T|W)$$

$$= \underset{T}{\text{argmax}} \frac{P_m(T)P_m(W|T)}{P_m(W)}$$

(Bayes' Theorem)

$$= \underset{T}{\text{argmax}} \, P_m(T)P_m(W|T)$$

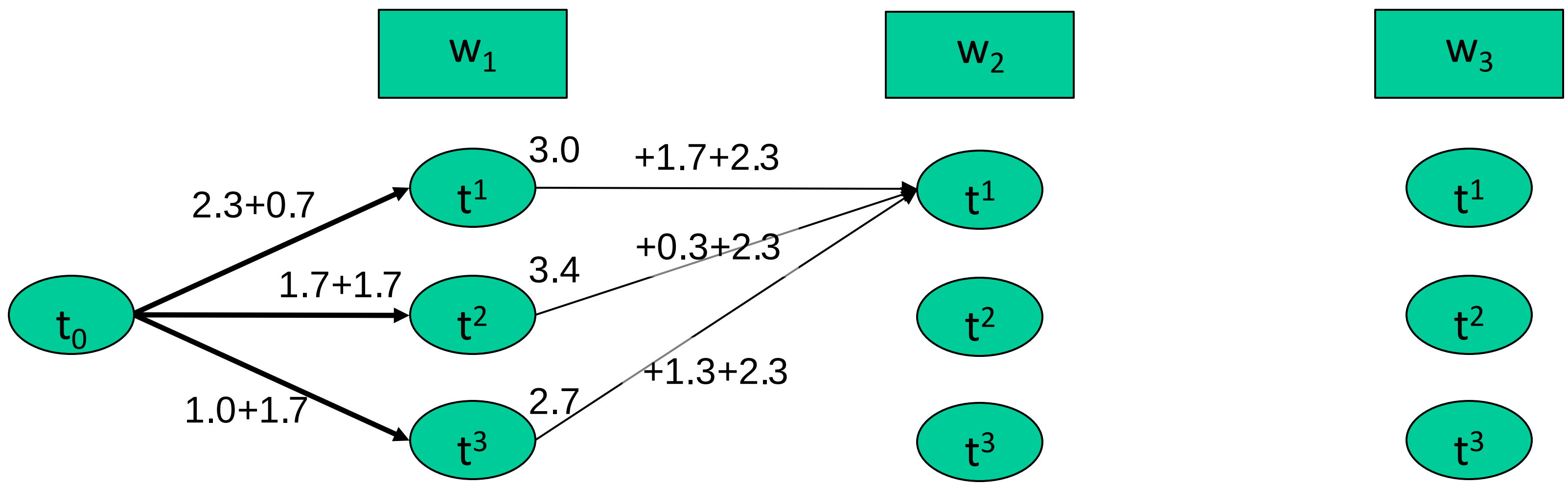
(W is constant for all T)

$$= \underset{T}{\text{argmax}} \prod_i (m(t_{i-1} \rightarrow t_i)m(w_i|t_i))$$

(First-order Markov assumption)

$$= \underset{T}{\text{argmax}} \sum_i \log(m(t_{i-1} \rightarrow t_i)m(w_i|t_i))$$

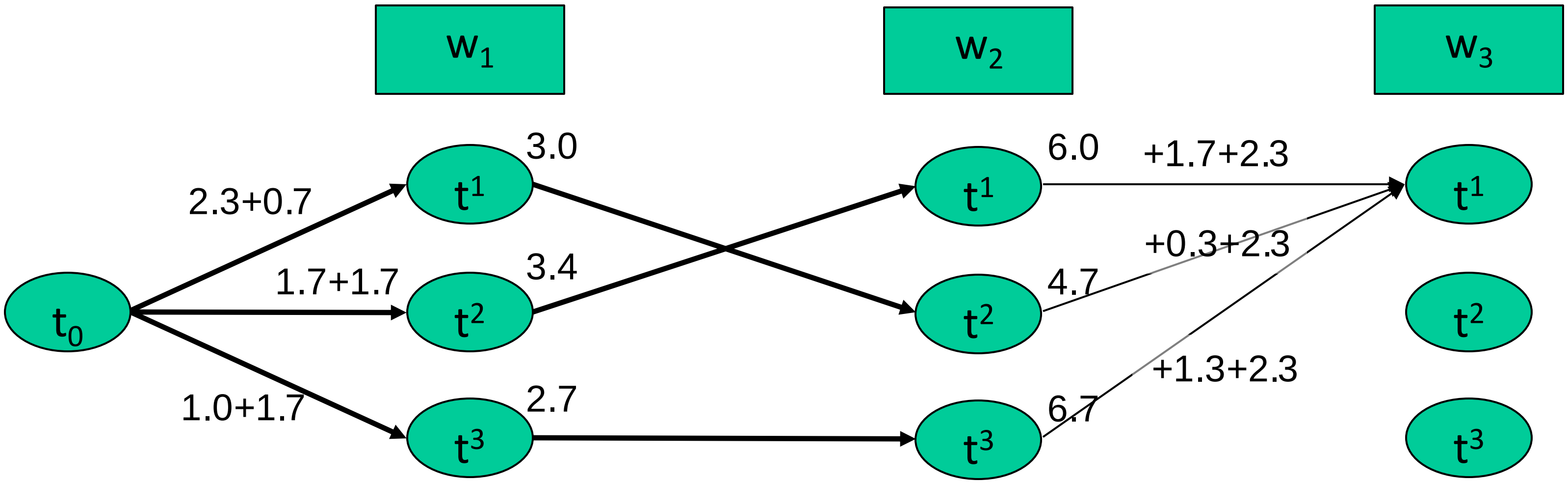
Viterbi algorithm



-log(m)	t^1	t^2	t^3
$t_0 \rightarrow$	2.3	1.7	1.0
$t^1 \rightarrow$	1.7	1.0	2.3
$t^2 \rightarrow$	0.3	3.3	3.3
$t^3 \rightarrow$	1.3	1.3	2.3

-log(m)	w^1	w^2	w^3
t^1	0.7	2.3	2.3
t^2	1.7	0.7	3.3
t^3	1.7	1.7	1.3

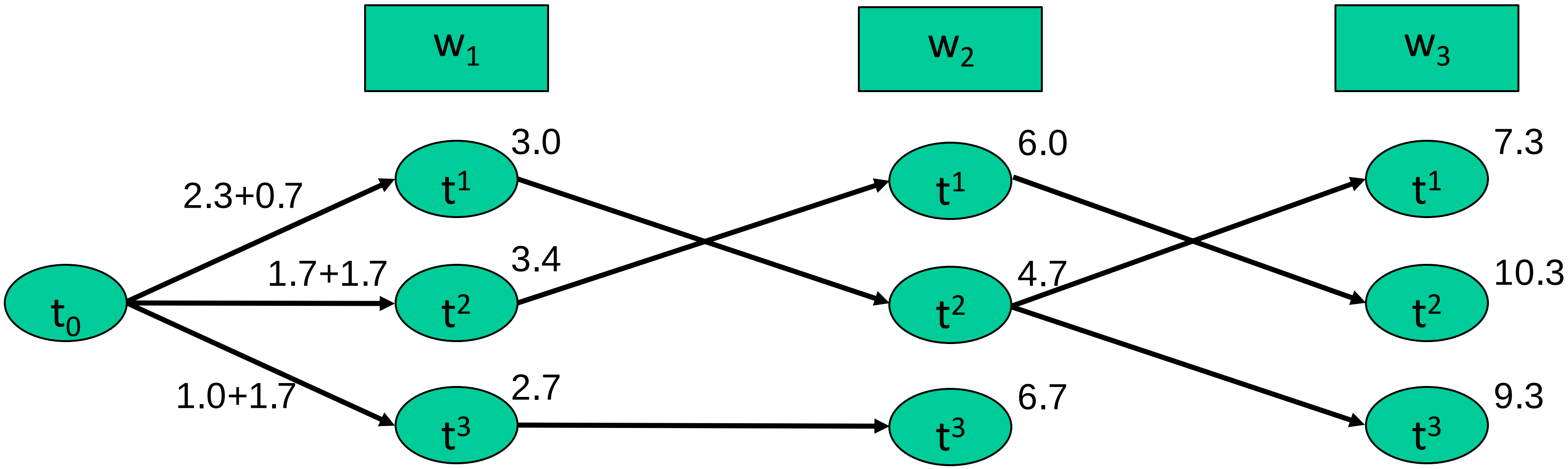
Viterbi algorithm



-log(m)	t^1	t^2	t^3
$t_0 \rightarrow$	2.3	1.7	1.0
$t^1 \rightarrow$	1.7	1.0	2.3
$t^2 \rightarrow$	0.3	3.3	3.3
$t^3 \rightarrow$	1.3	1.3	2.3

-log(m)	w^1	w^2	w^3
t^1	0.7	2.3	2.3
t^2	1.7	0.7	3.3
t^3	1.7	1.7	1.3

Viterbi algorithm



-log(m)	t^1	t^2	t^3
$t_0 \rightarrow$	2.3	1.7	1.0
$t^1 \rightarrow$	1.7	1.0	2.3
$t^2 \rightarrow$	0.3	3.3	3.3
$t^3 \rightarrow$	1.3	1.3	2.3

-log(m)	w^1	w^2	w^3
t^1	0.7	2.3	2.3
t^2	1.7	0.7	3.3
t^3	1.7	1.7	1.3

Viterbi Algorithm

```
D(0, START) = 0        ← P(t0 = Start) = 1
for each tag t != START do:
   D(0, t) = -∞        ← P(t0 <> Start) = 0
for i ← 1 to N do:
   for each tag tʲ do:
      D(i, tʲ) ← maxₖ (D(i-1,tᵏ)
                       + lm(wᵢ|tʲ)
                       + lm(tᵏ→tʲ))
log P(W,T) = maxⱼ D(N, tʲ)
```

where $lm(w_i|t^j) \stackrel{\text{def}}{=} \log P_m(w_i|t^j)$ and so forth

Viterbi Algorithm

```
D(0, START) = 0
for each tag t != START do:
  D(0, t) = -∞
for i ← 1 to N do:
  for each tag tʲ do:
     D(i, tʲ) ← maxₖ (D(i-1,tᵏ)
                         + lm(wᵢ|tʲ)
                         + lm(tᵏ→tʲ))
log P(W,T) = maxⱼ D(N, tʲ)
```

Score for each index, tag combination

where $lm(w_i|t^j) \stackrel{\text{def}}{=} \log P_m(w_i|t^j)$ and so forth

Viterbi Algorithm

```
D(0, START) = 0
for each tag t != START do:
  D(0, t) = -∞
for i ← 1 to N do:
  for each tag tʲ do:
     D(i, tʲ) ← maxₖ (D(i-1,tᵏ)
                       + lm(wᵢ|tʲ)
                       + lm(tᵏ→tʲ))
log P(W,T) = maxⱼ D(N, tʲ)
```

← Also want tag sequence that gives this max value

where $lm(w_i|t^j) \overset{\text{def}}{=} \log P_m(w_i|t^j)$ and so forth