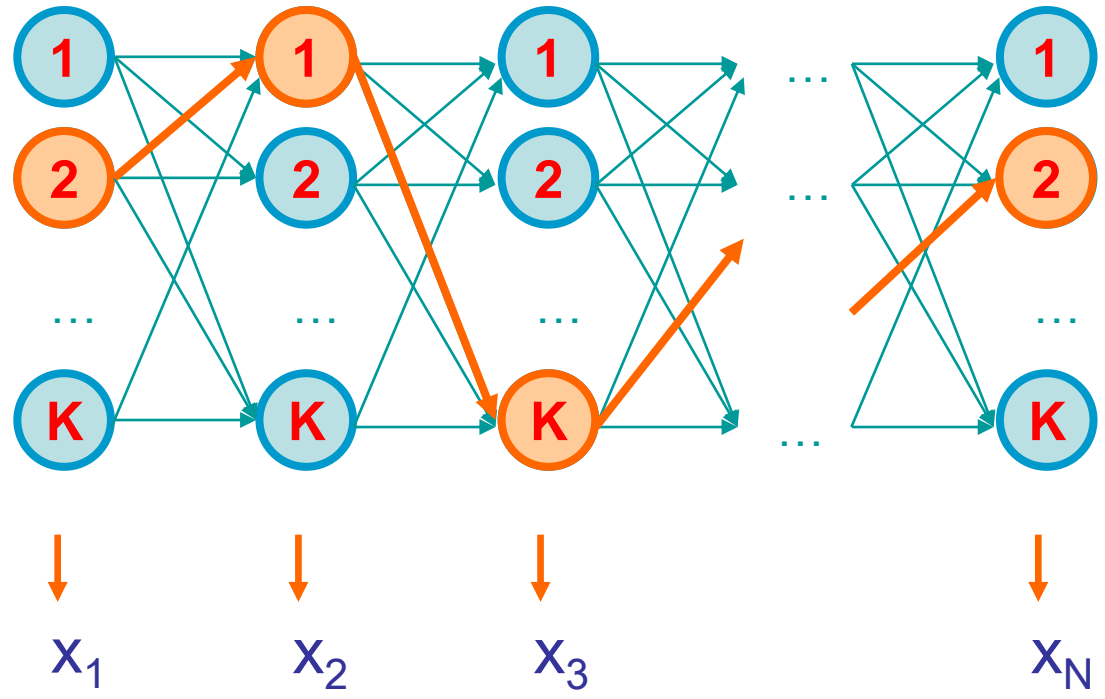




Hidden Markov Models





Example: The dishonest casino

A casino has two dice:

- Fair die

$$P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = 1/6$$

- Loaded die

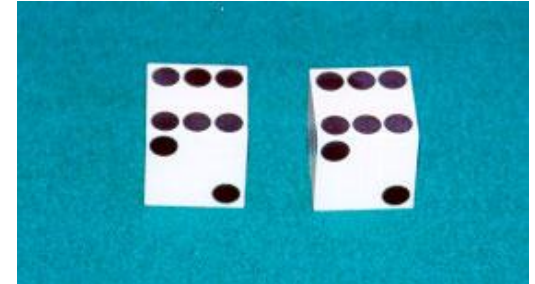
$$P(1) = P(2) = P(3) = P(4) = P(5) = 1/10$$

$$P(6) = 1/2$$

Casino player switches between fair and loaded die with probability $1/20$ at each turn

Game:

1. You bet \$1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins \$2





Question # 1 – Evaluation

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

Prob = 1.3×10^{-35}

QUESTION

How likely is this sequence, given our model of how the casino works?

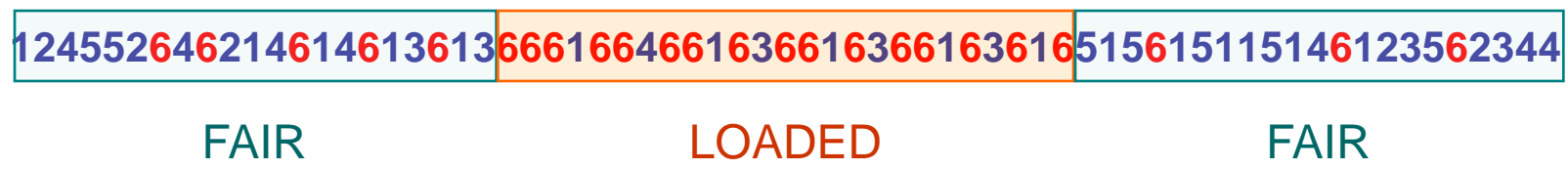
This is the **EVALUATION** problem in HMMs



Question # 2 – Decoding

GIVEN

A sequence of rolls by the casino player



QUESTION

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs



Question # 3 – Learning

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

Prob(6) = 64%

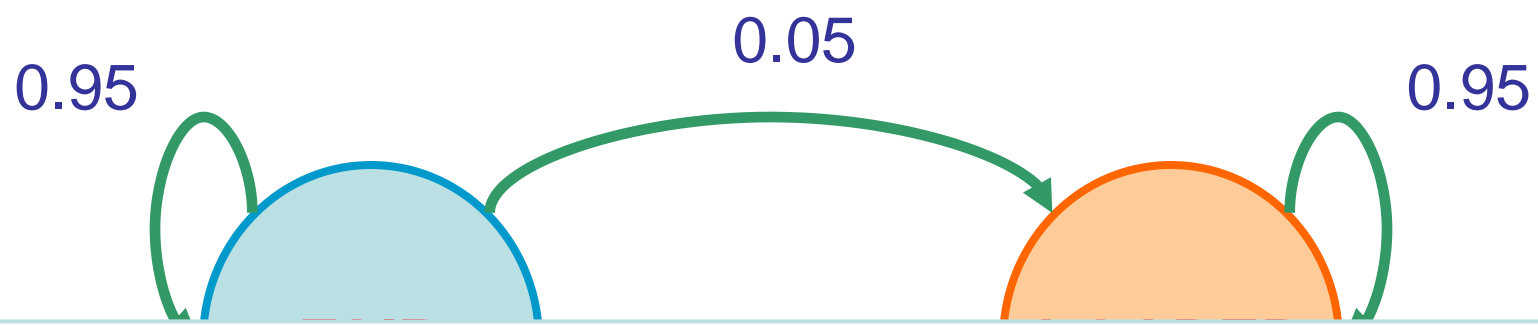
QUESTION

How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question in HMMs



The dishonest casino model



A *Hidden* Markov Model: we never observe the state, only observe **output** dependent (probabilistically) on state

- $P(3|F) = 1/6$
- $P(4|F) = 1/6$
- $P(5|F) = 1/6$
- $P(6|F) = 1/6$

0.05

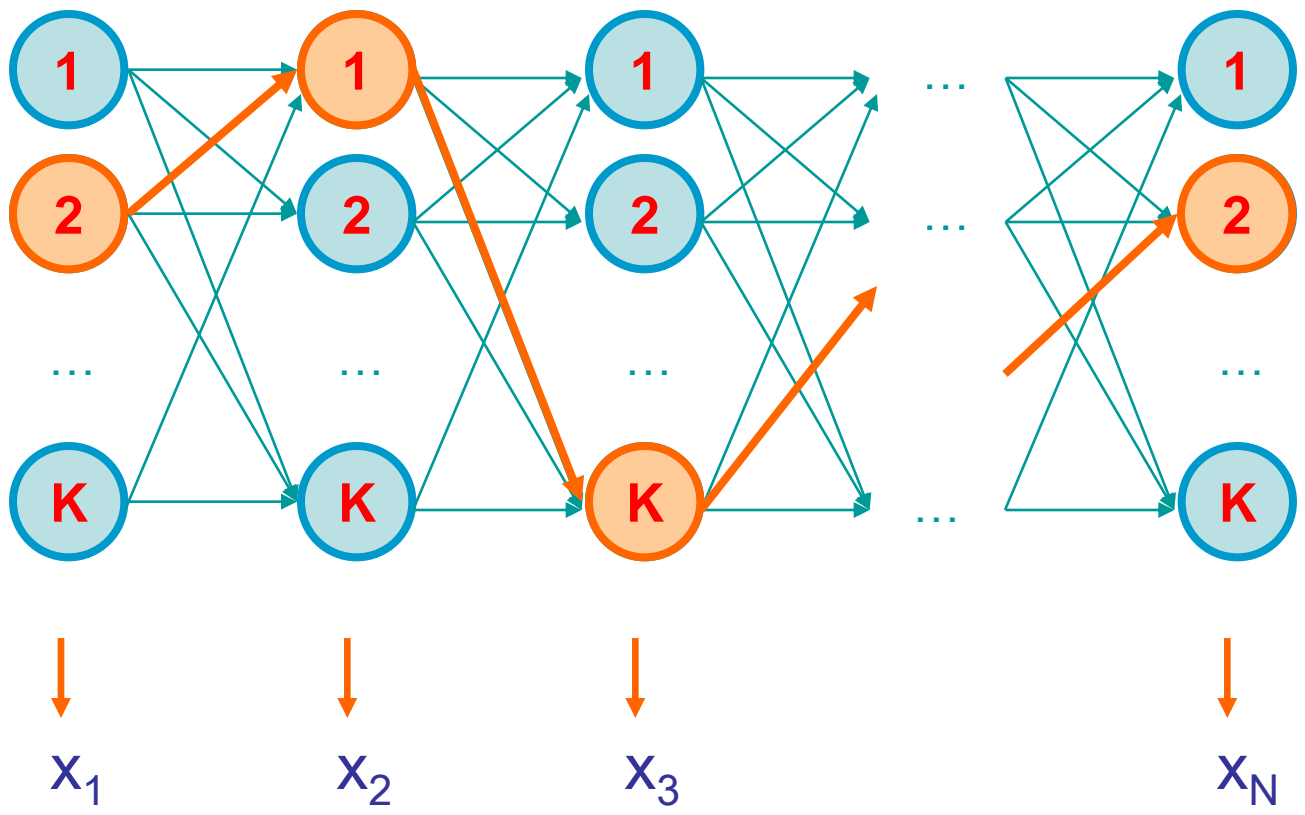
- $P(3|L) = 1/10$
- $P(4|L) = 1/10$
- $P(5|L) = 1/10$
- $P(6|L) = 1/2$



Sequences of states and observations

Observation sequence $\mathbf{x} = x_1 \dots x_N$,

State sequence $\boldsymbol{\pi} = \pi_1, \dots, \pi_N$

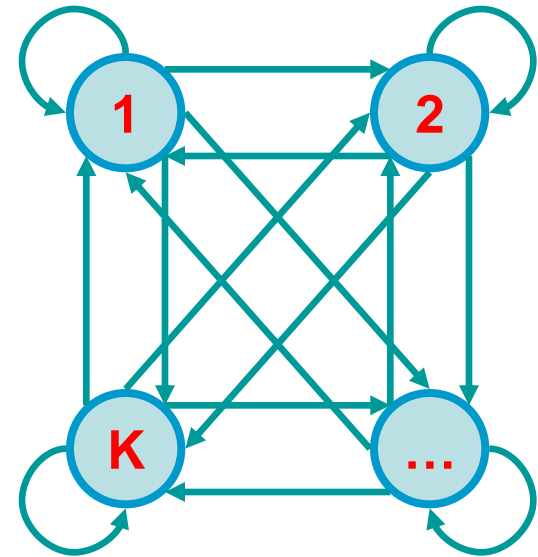


An HMM is memoryless



At each time step t ,
the only thing that affects future states
is the current state π_t

$$\begin{aligned} P(\pi_{t+1} = k \mid \text{“whatever happened so far”}) &= \\ P(\pi_{t+1} = k \mid \pi_1, \pi_2, \dots, \pi_t, x_1, x_2, \dots, x_t) &= \\ P(\pi_{t+1} = k \mid \pi_t) \end{aligned}$$

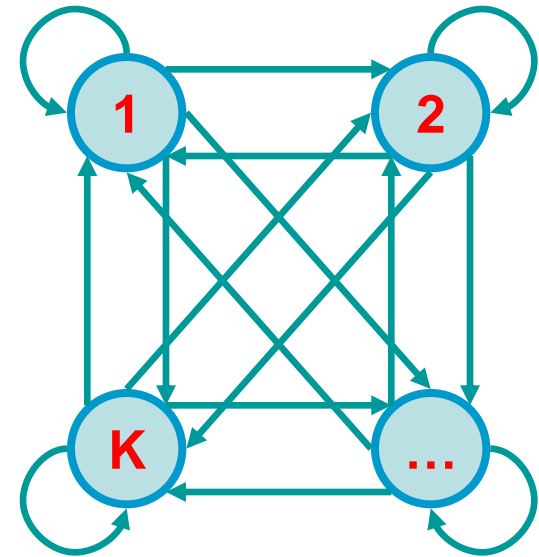


An HMM is memoryless



At each time step t ,
the only thing that affects x_t
is the current state π_t

$$\begin{aligned} P(x_t = b \mid \text{“whatever happened so far”}) &= \\ P(x_t = b \mid \pi_1, \pi_2, \dots, \pi_t, x_1, x_2, \dots, x_{t-1}) &= \\ P(x_t = b \mid \pi_t) \end{aligned}$$





Definition of a hidden Markov model

Definition: A hidden Markov model (HMM)

- **Alphabet** $\Sigma = \{ b_1, b_2, \dots, b_M \} = \text{Val}(x_t)$
- **Set of states** $Q = \{ 1, \dots, K \} \dots = \text{Val}(\pi_t)$
- **Transition probabilities** between any two states

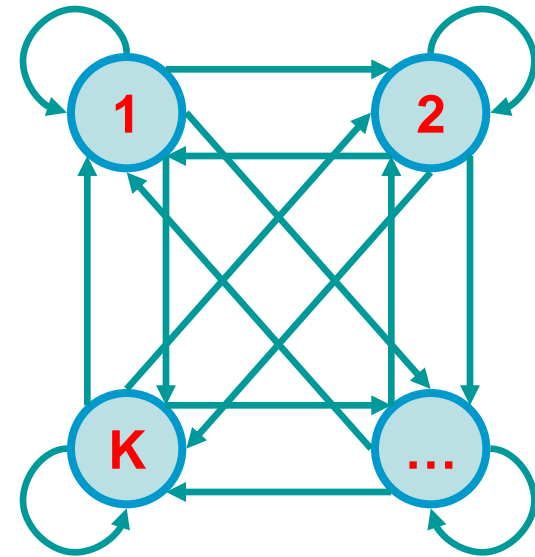
a_{ij} = transition prob from state i to state j
 $a_{i1} + \dots + a_{iK} = 1$, for all states $i = 1 \dots K$

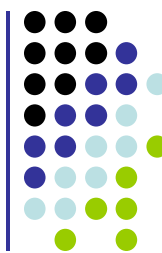
- **Start probabilities** a_{0i}

$a_{01} + \dots + a_{0K} = 1$

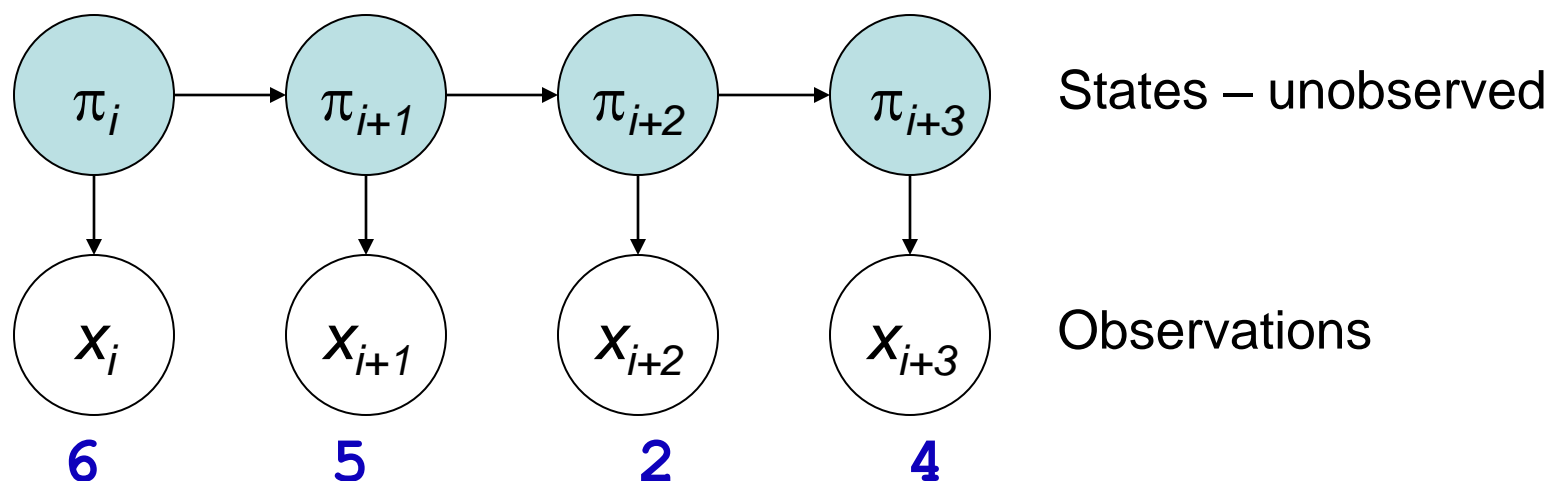
- **Emission probabilities** within each state

$e_k(b) = P(x_t = b \mid \pi_t = k)$
 $e_k(b_1) + \dots + e_k(b_M) = 1$, for all states $k = 1 \dots K$

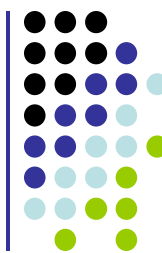




Hidden Markov Models as Bayes Nets



- **Transition probabilities** between any two states
 a_{jk} = transition prob from state j to state k
 $= P(\pi_{i+1} = k \mid \pi_i = j)$
- **Emission probabilities** within each state
 $e_k(b) = P(x_i = b \mid \pi_i = k)$



How “special” of a case are HMM BNs?

- *Template-based* representation
 - All hidden nodes have same CPTs, as do all output nodes
- Limited connectivity
 - Cliques of size ≤ 2
- Special-purpose algorithms are simple and efficient



HMMs are good for...

- Speech Recognition
- Gene Sequence Matching
- Text Processing
 - Part of speech tagging
 - Information extraction
 - Handwriting recognition

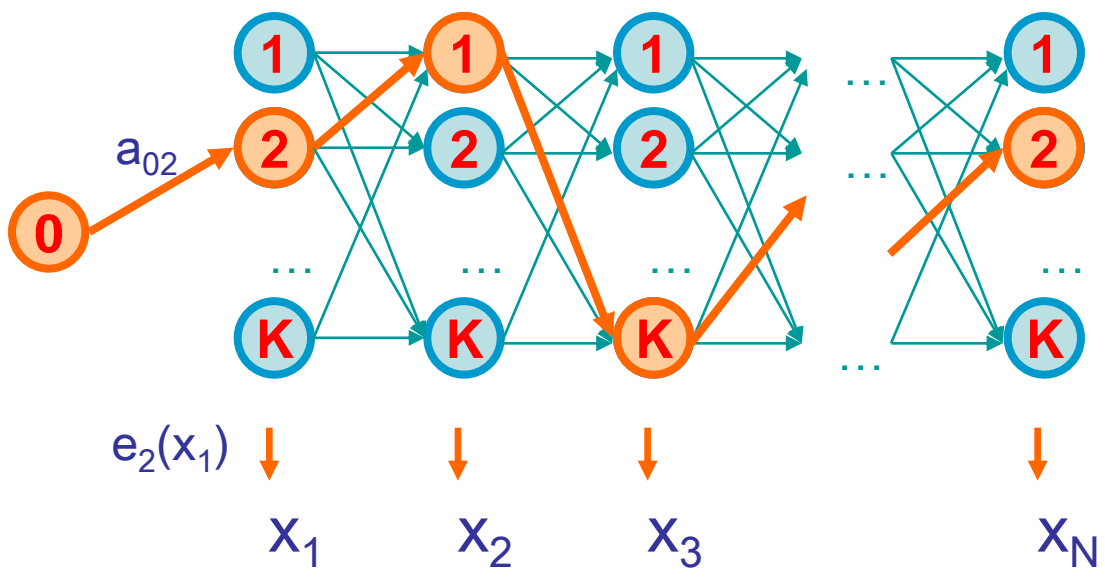
(i.e. not just dishonest casino applications)



Generating a sequence by the model

Given a HMM, we can generate a sequence of length n as follows:

1. Start at state π_1 according to prob $a_{0\pi_1}$
2. Emit letter x_1 according to prob $e_{\pi_1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi_1\pi_2}$
4. ... until emitting x_n

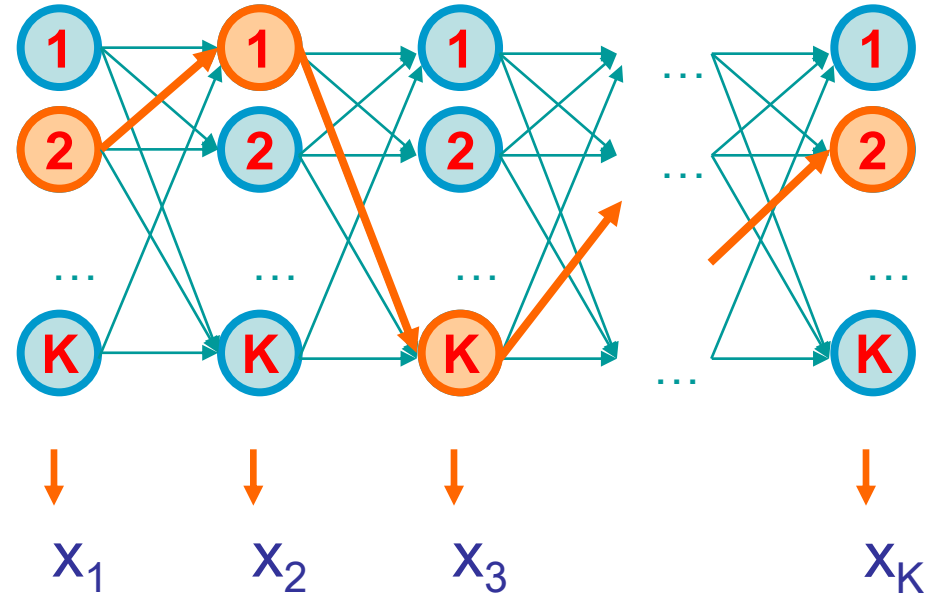




Likelihood of a parse

Given a sequence $\mathbf{x} = x_1 \dots x_N$
and a parse $\boldsymbol{\pi} = \pi_1, \dots, \pi_N$,

To find how likely this scenario is:
(given our HMM)



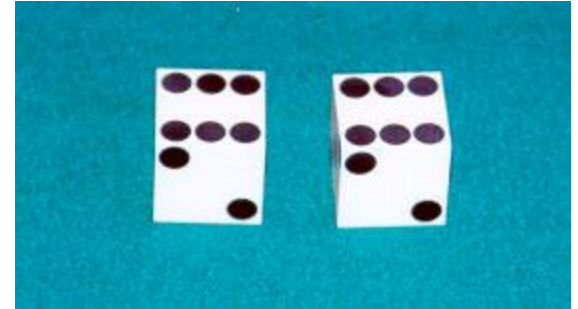
$$\begin{aligned} P(\mathbf{x}, \boldsymbol{\pi}) &= P(x_1, \dots, x_N, \pi_1, \dots, \pi_N) = \\ &P(x_N | \pi_N) P(\pi_N | \pi_{N-1}) \dots P(x_2 | \pi_2) P(\pi_2 | \pi_1) P(x_1 | \pi_1) P(\pi_1) = \\ &a_{0\pi_1} a_{\pi_1\pi_2} \dots a_{\pi_{N-1}\pi_N} e_{\pi_1}(x_1) \dots e_{\pi_N}(x_N) \end{aligned}$$



Example: the dishonest casino

Let the sequence of rolls be:

$$\mathbf{x} = 1, 2, 1, 5, 6, 2, 1, 5, 2, 4$$



Then, what is the likelihood of

$$\pi = \text{Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?}$$

(say initial probs $a_{0\text{Fair}} = 1/2$, $a_{0\text{Loaded}} = 1/2$)

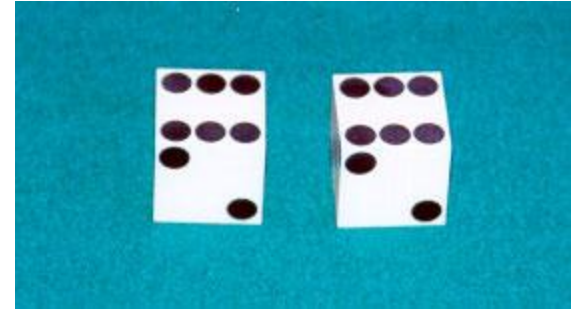
$$1/2 \times P(1 | \text{Fair}) P(\text{Fair} | \text{Fair}) P(2 | \text{Fair}) P(\text{Fair} | \text{Fair}) \dots P(4 | \text{Fair}) =$$

$$1/2 \times (1/6)^{10} \times (0.95)^9 = .00000000521158647211 \sim 0.5 \times 10^{-9}$$



Example: the dishonest casino

So, the likelihood the die is fair in this run is just 0.521×10^{-9}



What is the likelihood of

π = Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

$\frac{1}{2} \times P(1 \mid \text{Loaded}) P(\text{Loaded, Loaded}) \dots P(4 \mid \text{Loaded}) =$

$\frac{1}{2} \times (1/10)^9 \times (1/2)^1 (0.95)^9 = .00000000015756235243 \approx 0.16 \times 10^{-9}$

Therefore, it's somewhat more likely that all the rolls are done with the fair die, than that they are all done with the loaded die



Example: the dishonest casino

Let the sequence of rolls be:

$$\mathbf{x} = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6$$

Now, what is the likelihood $\pi = F, F, \dots, F$?

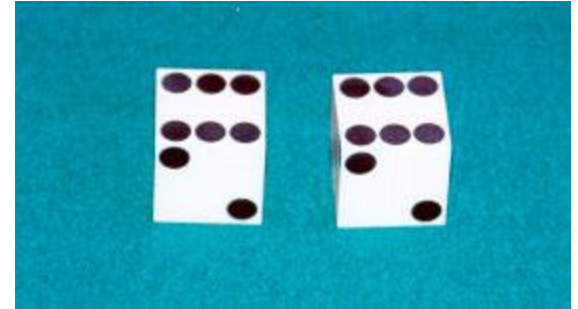
$$\frac{1}{2} \times (1/6)^{10} \times (0.95)^9 \approx 0.5 \times 10^{-9}, \text{ same as before}$$

What is the likelihood

$$\pi = L, L, \dots, L?$$

$$\frac{1}{2} \times (1/10)^4 \times (1/2)^6 (0.95)^9 = .00000049238235134735 \approx 0.5 \times 10^{-7}$$

So, it is 100 times more likely the die is loaded





The three main questions for HMMs

1. Evaluation

GIVEN a HMM M , and a sequence \mathbf{x} ,
FIND $\text{Prob}[\mathbf{x} | M]$

2. Decoding

GIVEN a HMM M , and a sequence \mathbf{x} ,
FIND the sequence π of states that maximizes $P[\mathbf{x}, \pi | M]$

3. Learning

GIVEN a HMM M , with unspecified transition/emission probs.,
and a sequence \mathbf{x} ,
FIND parameters $\theta = (e_i(\cdot), a_{ij})$ that maximize $P[\mathbf{x} | \theta]$



Problem 1: Evaluation

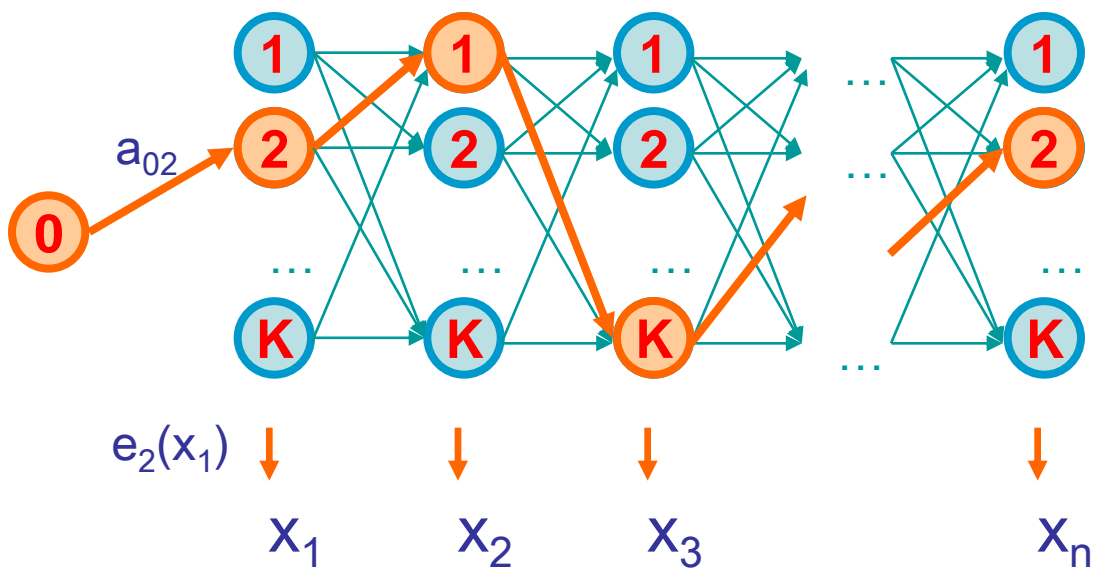
Compute the likelihood that a sequence is generated by the model



Generating a sequence by the model

Given a HMM, we can generate a sequence of length n as follows:

1. Start at state π_1 according to prob $a_{0\pi_1}$
2. Emit letter x_1 according to prob $e_{\pi_1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi_1\pi_2}$
4. ... until emitting x_n





Evaluation

We will develop algorithms that allow us to compute:

$P(\mathbf{x})$ Probability of \mathbf{x} given the model

$P(x_i \dots x_j)$ Probability of a substring of \mathbf{x} given the model

$P(\pi_i = k \mid \mathbf{x})$ “**Posterior**” probability that the i^{th} state is k , given \mathbf{x}



The Forward Algorithm

We want to calculate

$P(\mathbf{x})$ = probability of \mathbf{x} , given the HMM

Sum over all possible ways of generating \mathbf{x} :

$$P(\mathbf{x}) = \sum_{\pi} P(\mathbf{x}, \pi) = \sum_{\pi} P(\mathbf{x} | \pi) P(\pi)$$

To avoid summing over exponentially many paths π , use *variable elimination*. Define:

$$f_k(i) = P(x_1 \dots x_i, \pi_i = k) \quad (\text{the forward probability})$$

“generate first i observations and end up in state k ”



The Forward Algorithm – derivation

Define the forward probability:

$$\begin{aligned} f_k(i) &= P(x_1 \dots x_i, \pi_i = k) \\ &= \sum_{\pi_1 \dots \pi_{i-1}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, \pi_i = k) e_k(x_i) \\ &= \sum_j \sum_{\pi_1 \dots \pi_{i-2}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-2}, \pi_{i-1} = j) a_{jk} e_k(x_i) \\ &= \sum_j P(x_1 \dots x_{i-1}, \pi_{i-1} = j) a_{jk} e_k(x_i) \\ &= e_k(x_i) \sum_j f_j(i-1) a_{jk} \end{aligned}$$



The Forward Algorithm

We can compute $f_k(i)$ for all k, i , using dynamic programming!

Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$f_k(i) = e_k(x_i) \sum_j f_j(i-1) a_{jk}$$

Termination:

$$P(\mathbf{x}) = \sum_k f_k(N)$$



Backward Algorithm

Forward algorithm can compute $P(\mathbf{x})$. But we'd also like:

$$P(\pi_i = k \mid \mathbf{x}),$$

the probability distribution on the i^{th} position, given \mathbf{x}

We start by computing:

$$\begin{aligned} P(\pi_i = k, \mathbf{x}) &= P(x_1 \dots x_i, \pi_i = k, x_{i+1} \dots x_N) \\ &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_N \mid x_1 \dots x_i, \pi_i = k) \\ &= \boxed{P(x_1 \dots x_i, \pi_i = k)} \boxed{P(x_{i+1} \dots x_N \mid \pi_i = k)} \end{aligned}$$

Forward, $f_k(i)$ **Backward, $b_k(i)$**

Then, $P(\pi_i = k \mid \mathbf{x}) = P(\pi_i = k, \mathbf{x}) / P(\mathbf{x})$



The Backward Algorithm – derivation

Define the backward probability:

$$\begin{aligned} b_k(i) &= P(x_{i+1} \dots x_N \mid \pi_i = k) && \text{“starting from } i^{\text{th}} \text{ state = } k, \text{ generate rest of } x\text{”} \\ &= \sum_{\pi_{i+1} \dots \pi_N} P(x_{i+1}, x_{i+2}, \dots, x_N, \pi_{i+1}, \dots, \pi_N \mid \pi_i = k) \\ &= \sum_j \sum_{\pi_{i+2} \dots \pi_N} P(x_{i+1}, x_{i+2}, \dots, x_N, \pi_{i+1} = j, \pi_{i+2}, \dots, \pi_N \mid \pi_i = k) \\ &= \sum_j e_j(x_{i+1}) a_{kj} \sum_{\pi_{i+2} \dots \pi_N} P(x_{i+2}, \dots, x_N, \pi_{i+2}, \dots, \pi_N \mid \pi_{i+1} = j) \\ &= \sum_j e_j(x_{i+1}) a_{kj} \mathbf{b_j(i+1)} \end{aligned}$$



The Backward Algorithm

We can compute $b_k(i)$ for all k, i , using dynamic programming

Initialization:

$$b_k(N) = 1, \text{ for all } k$$

Iteration:

$$b_k(i) = \sum_l e_l(x_{i+1}) a_{kl} b_l(i+1)$$

Termination:

$$P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$$



Computational Complexity

What is the running time, and space required, for Forward and Backward?

Time: $O(K^2N)$
Space: $O(KN)$

Useful implementation technique to avoid underflows:

rescaling at each few positions by multiplying
by a constant



Posterior Decoding

We can now calculate

$$P(\pi_i = k | x) = \frac{f_k(i) b_k(i)}{P(x)}$$

Then, we can ask

$$P(\pi_i = k | x) =$$

$$P(\pi_i = k, x) / P(x) =$$

$$P(x_1, \dots, x_i, \pi_i = k, x_{i+1}, \dots, x_n) / P(x) =$$

$$P(x_1, \dots, x_i, \pi_i = k) P(x_{i+1}, \dots, x_n | \pi_i = k) / P(x) =$$

$$f_k(i) b_k(i) / P(x)$$

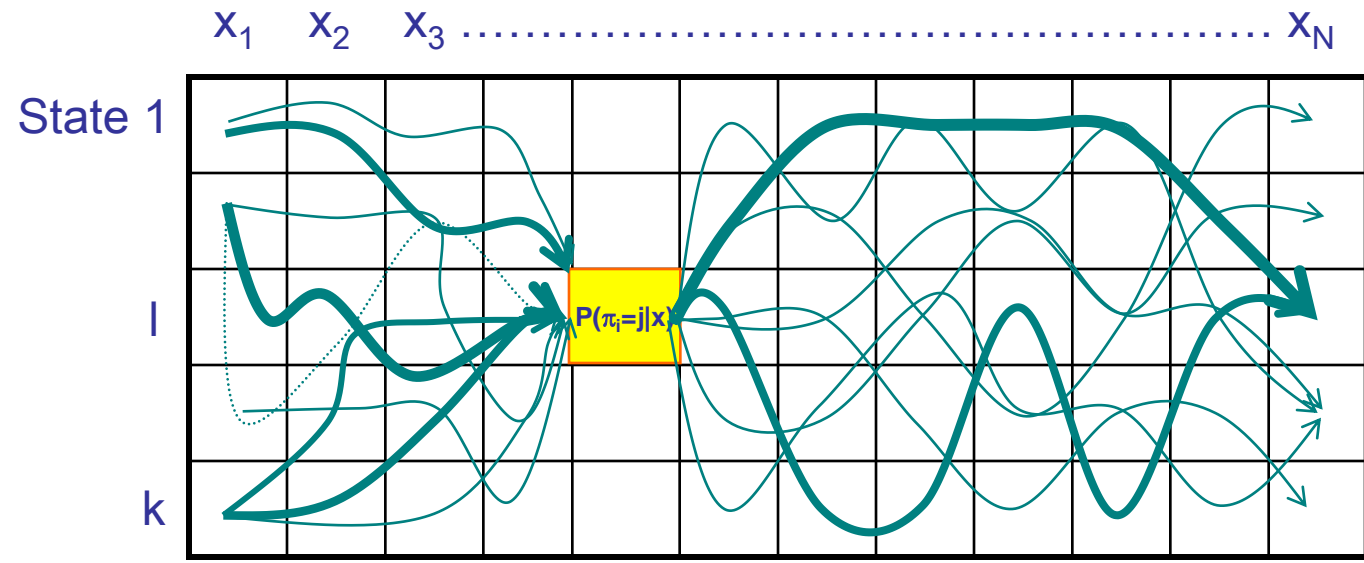
What is the most likely state at position i of sequence x :

Define $\hat{\pi}_i$ by Posterior Decoding:

$$\hat{\pi}_i = \operatorname{argmax}_k P(\pi_i = k | x)$$



Posterior Decoding





Problem 2: Decoding

*Find the most likely parse
of a sequence*



Most likely sequences vs. ind. states

Given a sequence \mathbf{x} ,

- Most likely *sequence* of states
- Most likely *states*

$$\begin{aligned}
 P(\text{box: FFFFFFFFFFFF}) &= \\
 (1/6)^{11} * 0.95^{12} &= \\
 2.76 \cdot 10^{-9} * 0.54 &= \\
 1.49 \cdot 10^{-9} &
 \end{aligned}$$

$$\begin{aligned}
 P(\text{box: LLLLLLLLLLLL}) &= \\
 [(1/2)^6 * (1/10)^5] * 0.95^{10} * 0.05^2 &= \\
 1.56 \cdot 10^{-7} * 1.5 \cdot 10^{-3} &= \\
 0.23 \cdot 10^{-9} &
 \end{aligned}$$

Example: the dishonest card

Say $x = 12341 \dots 231 \mathbf{62616364616} 234112 \dots 21341$

$\underbrace{\hspace{10em}}$
 $\underbrace{\hspace{10em}}$

F
F

Most likely path: $\pi = FF \dots F$

(too “unlikely” to transition $F \rightarrow L \rightarrow F$)

However: marked letters more likely to be L than unmarked letters



Decoding

GIVEN $\mathbf{x} = x_1 x_2 \dots x_N$

Find $\pi = \pi_1, \dots, \pi_N$,
to maximize $P[\mathbf{x}, \pi]$

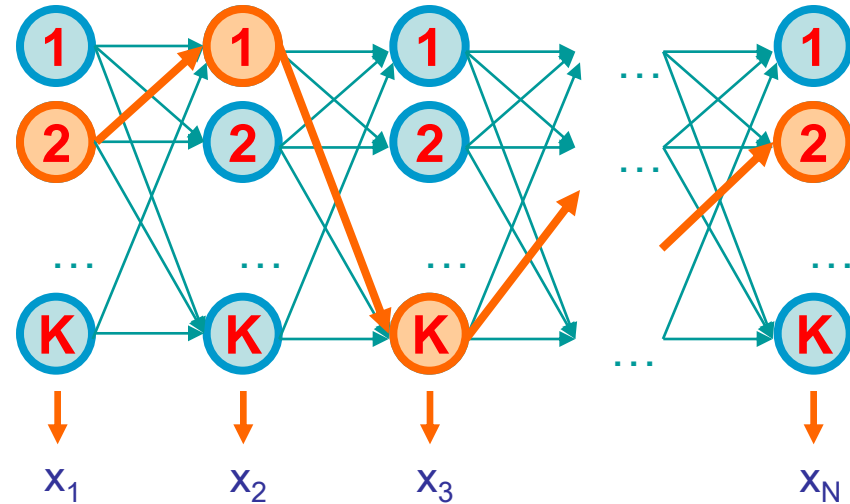
$$\pi^* = \operatorname{argmax}_{\pi} P[\mathbf{x}, \pi]$$

Maximizes $a_{0\pi_1} e_{\pi_1}(x_1) a_{\pi_1\pi_2} \dots a_{\pi_{N-1}\pi_N} e_{\pi_N}(x_N)$

Like forward alg, with max instead of sum

$$V_k(i) = \max_{\{\pi_1 \dots \pi_{i-1}\}} P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i = k]$$

= Prob. of most likely sequence of states ending at state $\pi_i = k$





Decoding – main idea

Induction: Given that for all states k , and for a fixed position i ,

$$V_k(i) = \max_{\{\pi_1 \dots \pi_{i-1}\}} P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i = k]$$

What is $V_j(i+1)$?

From definition,

$$\begin{aligned}
V_j(i+1) &= \max_{\{\pi_1 \dots \pi_i\}} P[x_1 \dots x_i, \pi_1, \dots, \pi_i, x_{i+1}, \pi_{i+1} = j] \\
&= \max_{\{\pi_1 \dots \pi_i\}} P(x_{i+1}, \pi_{i+1} = j \mid x_1 \dots x_i, \pi_1, \dots, \pi_i) P[x_1 \dots x_i, \pi_1, \dots, \pi_i] \\
&= \max_{\{\pi_1 \dots \pi_i\}} P(x_{i+1}, \pi_{i+1} = j \mid \pi_i) P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i] \\
&= \max_k [P(x_{i+1}, \pi_{i+1} = j \mid \pi_i = k) \max_{\{\pi_1 \dots \pi_{i-1}\}} P[x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i = k]] \\
&= \max_k [P(x_{i+1} \mid \pi_{i+1} = j) P(\pi_{i+1} = j \mid \pi_i = k) V_k(i)] \\
&= e_j(x_{i+1}) \max_k a_{kj} V_k(i)
\end{aligned}$$



The Viterbi Algorithm

Input: $x = x_1 \dots x_N$

Initialization:

$$V_0(0) = 1$$

(0 is the imaginary first position)

$$V_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$V_j(i) = e_j(x_i) \times \max_k a_{kj} V_k(i-1)$$

$$\text{Ptr}_j(i) = \text{argmax}_k a_{kj} V_k(i-1)$$

Termination:

$$P(x, \pi^*) = \max_k V_k(N)$$

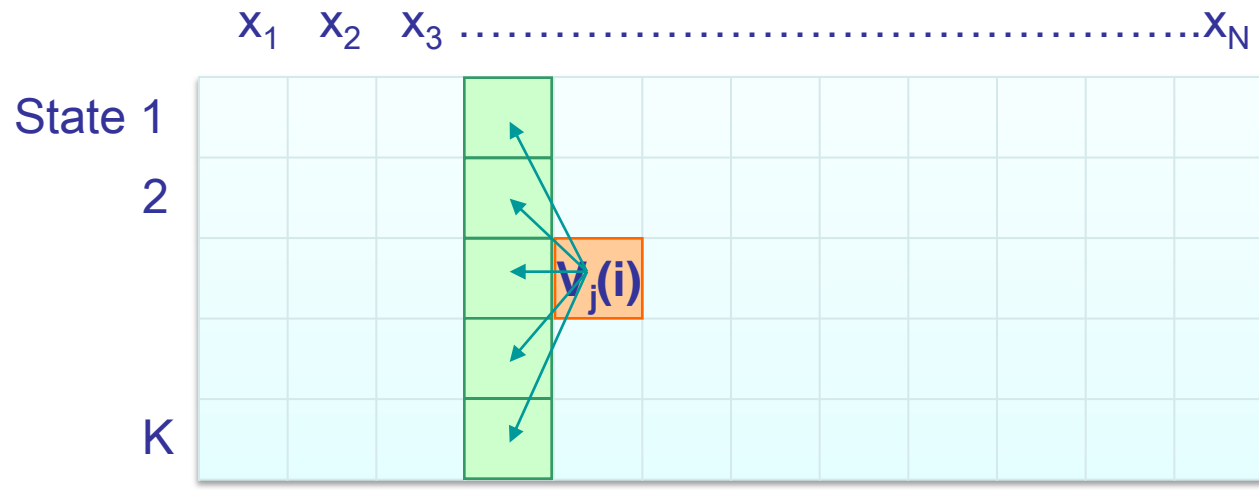
Traceback:

$$\pi_N^* = \text{argmax}_k V_k(N)$$

$$\pi_{i-1}^* = \text{Ptr}_{\pi_i^*}(i)$$



The Viterbi Algorithm



Time: $O(K^2N)$

Space: $O(KN)$



Viterbi Algorithm – a practical detail

Underflows are a significant problem (like with forward, backward)

$$P[\mathbf{x}_1, \dots, \mathbf{x}_i, \pi_1, \dots, \pi_i] = a_{0\pi_1} a_{\pi_1\pi_2} \dots a_{\pi_i} e_{\pi_1}(\mathbf{x}_1) \dots e_{\pi_i}(\mathbf{x}_i)$$

These numbers become extremely small – underflow

Solution: Take the logs of all values

$$V_i(i) = \log e_k(\mathbf{x}_i) + \max_k [V_k(i-1) + \log a_{ki}]$$



Example

Let x be a long sequence with a portion of $\sim 1/6$ 6's,
followed by a portion of $\sim 1/2$ 6's...

$x = 123456123456\dots123456$ $6626364656\dots1626364656$

Then, it is not hard to show that optimal parse is:

$FFF\dots F$ $LLL\dots L$

6 characters "123456" parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$
parsed as L, contribute $.95^6 \times (1/2)^1 \times (1/10)^5 = 0.4 \times 10^{-5}$

"162636" parsed as F, contribute $.95^6 \times (1/6)^6 = 1.6 \times 10^{-5}$
parsed as L, contribute $.95^6 \times (1/2)^3 \times (1/10)^3 = 9.0 \times 10^{-5}$



Viterbi, Forward, Backward

VITERBI

Initialization:

$$V_0(0) = 1$$

$$V_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$V_l(i) = e_l(x_i) \max_k V_k(i-1) a_{kl}$$

Termination:

$$P(x, \pi^*) = \max_k V_k(N)$$

FORWARD

Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \text{ for all } k > 0$$

Iteration:

$$f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$$

Termination:

$$P(x) = \sum_k f_k(N)$$

BACKWARD

Initialization:

$$b_k(N) = 1, \text{ for all } k$$

Iteration:

$$b_l(i) = \sum_k e_l(x_{i+1}) a_{kl} b_k(i+1)$$

Termination:

$$P(x) = \sum_k a_{0k} e_k(x_1) b_k(1)$$



Problem 3: Learning

Find the parameters that maximize the likelihood of the observed sequence



Estimating HMM parameters

- Easy if we know the sequence of hidden states
 - Count # times each transition occurs
 - Count #times each observation occurs in each state
- Given an HMM and observed sequence, we can compute the distribution over paths, and therefore the expected counts
- “Chicken and egg” problem



Solution: Use the EM algorithm

- Guess initial HMM parameters
- **E step:** Compute distribution over paths
- **M step:** Compute max likelihood parameters
- But how do we do this efficiently?

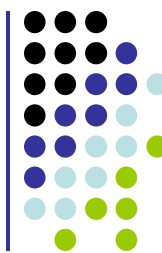


The forward-backward algorithm

- Also known as the Baum-Welch algorithm
- Compute probability of each state at each position using forward and backward probabilities
→ (Expected) observation counts
- Compute probability of each pair of states at each pair of consecutive positions i and $i+1$ using $forward(i)$ and $backward(i+1)$
→ (Expected) transition counts

$$\text{Count}(k \rightarrow l) = \sum_i f_k(i) a_{kl} b_l(i+1) / P(x)$$

Application: HMMs for Information Extraction (IE)



- IE: Text → machine-understandable data

Paris, the capital of **France**, ...



(**Paris**, **France**) ∈ CapitalOf, $p=0.85$

- Applied to Web: better search engines, semantic Web, step toward human-level AI



IE Automatically?

Intractable to get human labels for every concept expressed on the Web

Idea: extract from **semantically tractable** sentences

...Edison **invented** the light bulb...

(Edison, light bulb) \in **Invented**

$$\mathbf{x} \mathbf{V} \mathbf{y} \Rightarrow (\mathbf{x}, \mathbf{y}) \in \mathbf{V}$$

...Bloomberg, **mayor** of New York City...

\Rightarrow (Bloomberg, New York City) \in **Mayor**

$$\mathbf{x}, \mathbf{C} \text{ of } \mathbf{y} \Rightarrow (\mathbf{x}, \mathbf{y}) \in \mathbf{C}$$

But...

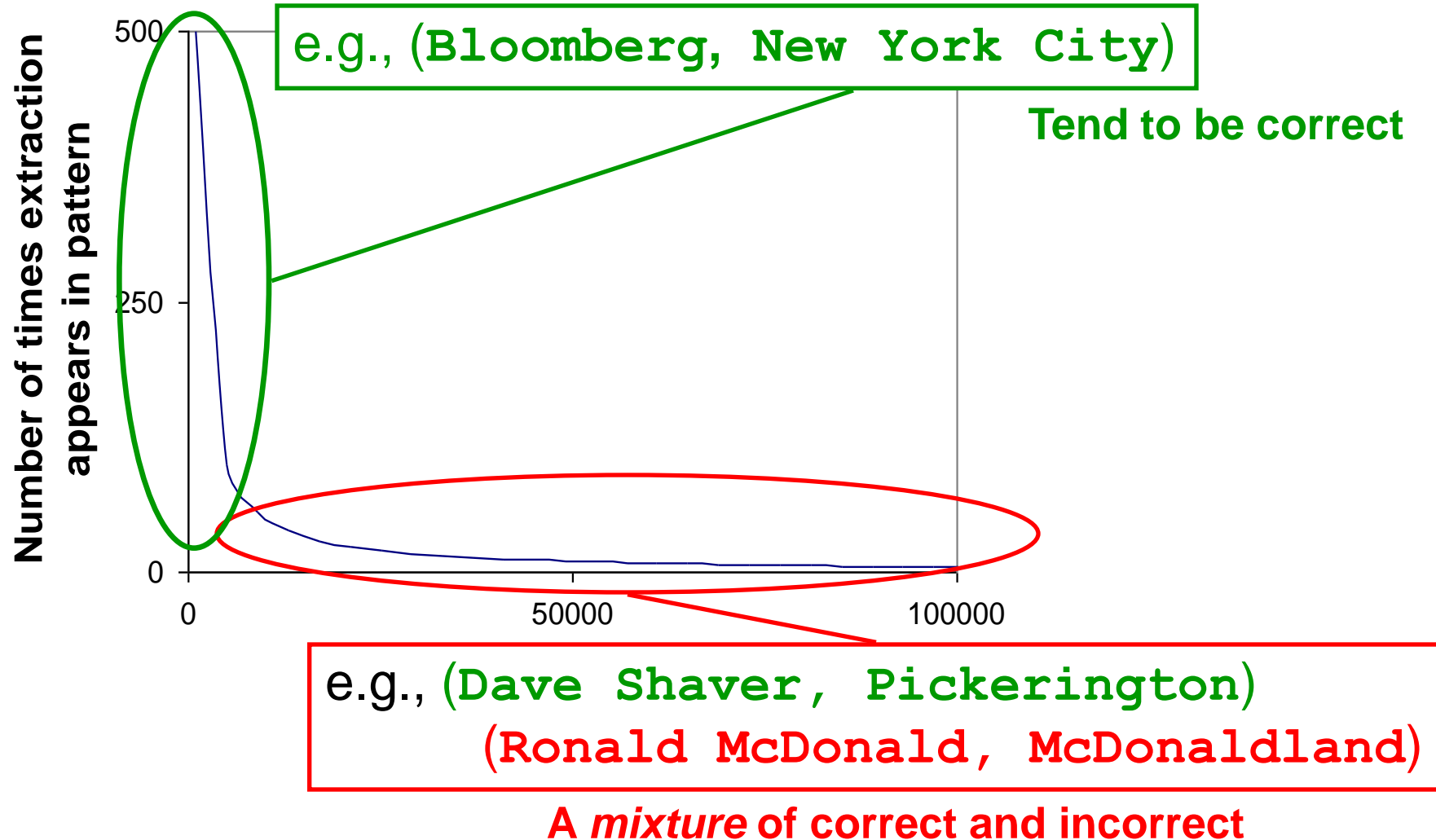


Extraction patterns make errors:

“Erik Jonsson, CEO of **Texas Instruments**,
mayor of **Dallas** from 1964-1971, and...”

- Empirical fact:
 - Extractions you see over and over tend to be correct
 - The problem is the “long tail”

Challenge: the “long tail”



Mayor McCheese





Assessing Sparse Extractions

Strategy

- 1) Model how **common** extractions occur in text
- 2) Rank **sparse** extractions by fit to model

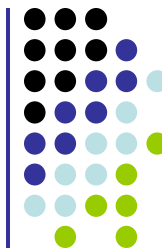


The Distributional Hypothesis

- *Terms in the same class tend to appear in similar contexts.*

Context	Hits with Chicago	Hits with Twisp
"cities including ___"	42,000	1
"___ and other cities"	37,900	0
"___ hotels"	2,000,000	1,670
"mayor of ___"	657,000	82

HMM Language Models

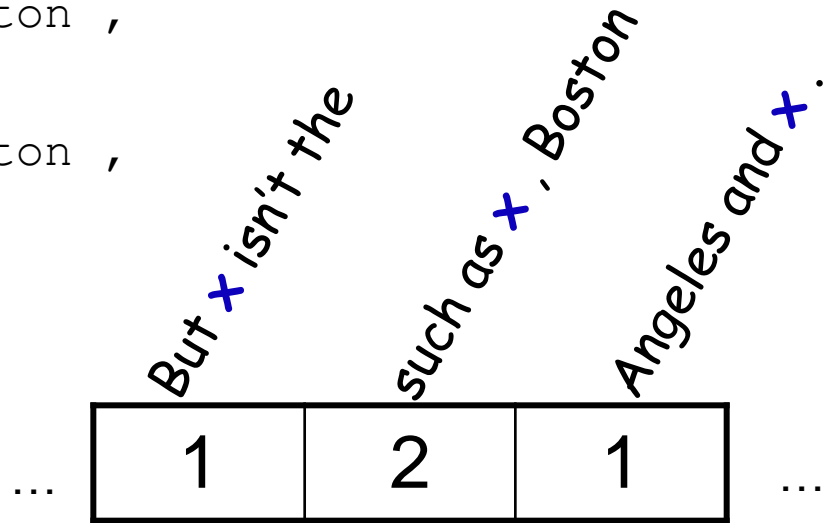


- Precomputed – scalable
- Handle sparsity



Baseline: context vectors

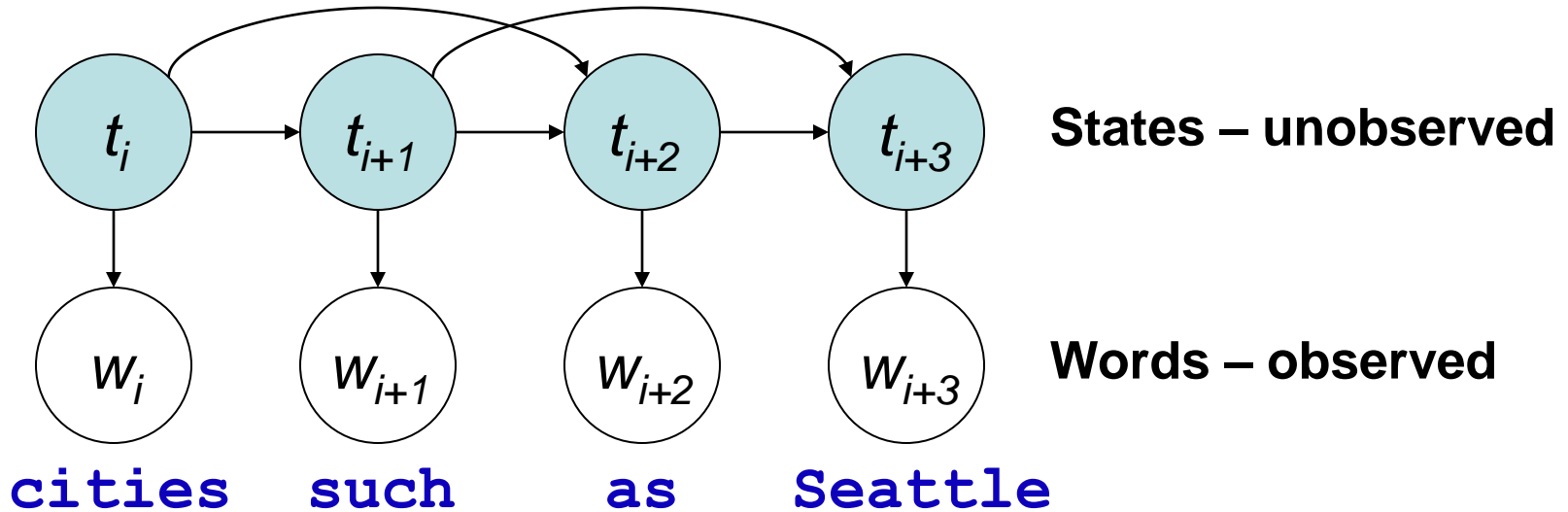
...
cities such as **Chicago** , Boston ,
But **Chicago** isn't the best
cities such as **Chicago** , Boston ,
Los Angeles and **Chicago** .
...



- Compute dot products between vectors of common and sparse extractions
[\[cf. Ravichandran et al. 2005\]](#)



Hidden Markov Model (HMM)

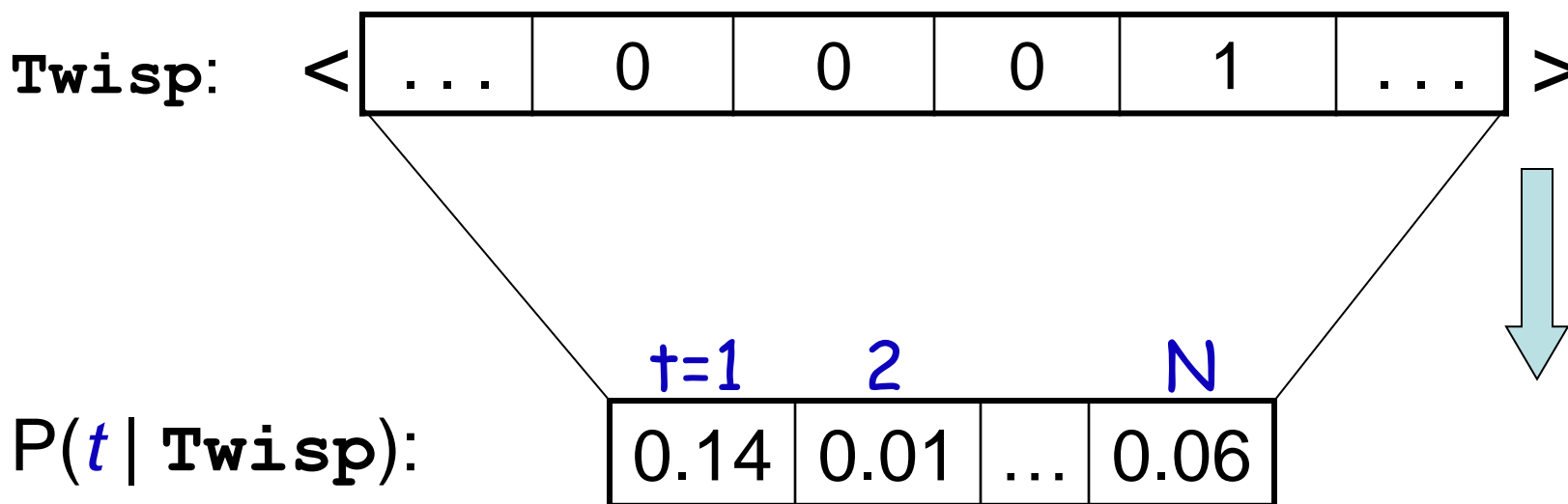
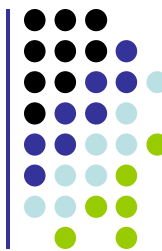


Hidden States $t_j \in \{1, \dots, N\}$ (N fairly small)

Train on **unlabeled** data

- $P(t_j \mid w_j = w)$ is N -dim. **distributional summary** of w
- Compare extractions using KL divergence

HMM Compresses Context Vectors



Distributional Summary $P(t \mid w)$

- Compact (efficient – **10-50x** less data retrieved)
- Dense (accurate – **23-46%** error reduction)



Example

Is **Pickerington** of the same type as **Chicago**?

Chicago , Illinois
Pickerington , Ohio

Chicago:

Pickerington:

291	0	...
0	1	...

$\langle x \rangle$, Illinois

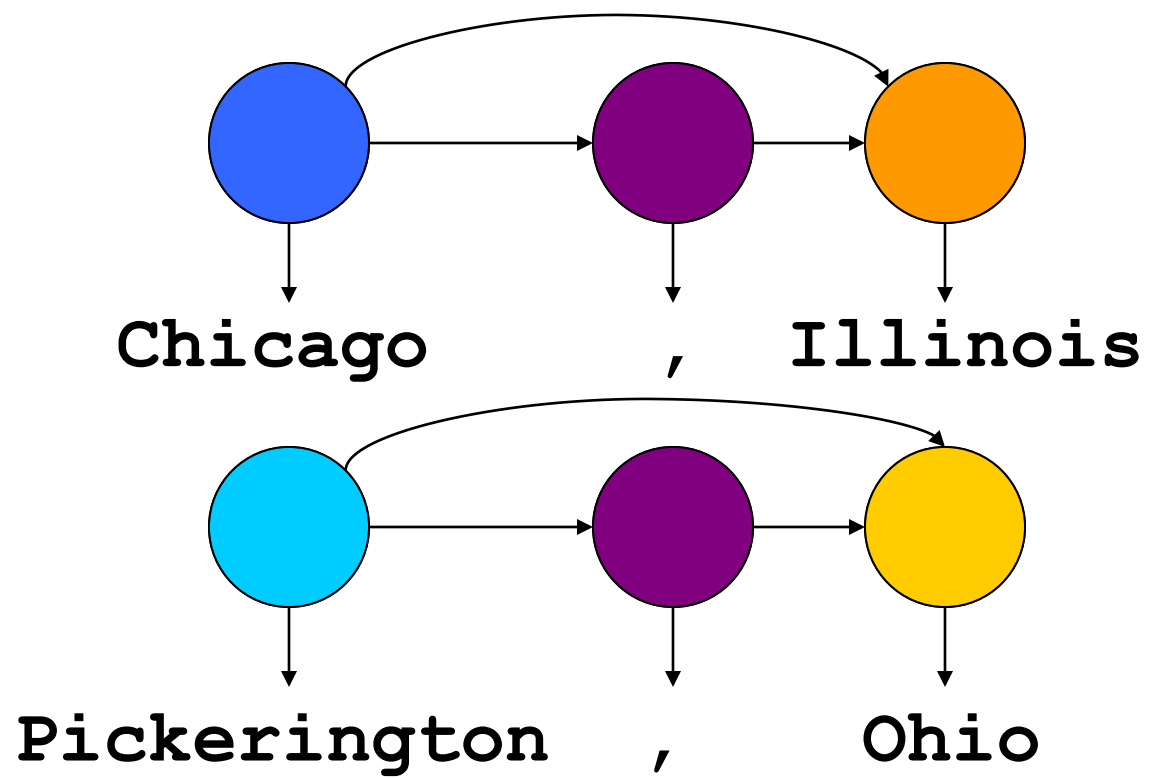
$\langle x \rangle$, Ohio

=> Context vectors say **no**, dot product is 0!



Example

HMM Generalizes:





Experimental Results

Task: Ranking sparse TextRunner extractions.

Metric: Area under precision-recall curve.

	Headquartered	Merged		Average
Frequency	0.710	0.784		0.713
PL	0.651	0.851	...	0.785
LM	0.810	0.908		0.851

Language models reduce missing area by **39%** over nearest competitor.



Example word distributions (1 of 2)

- P(word | state 3)

- unk0 0.0244415
- new 0.0235757
- more 0.0123496
- unk1 0.0119841
- few 0.0114422
- small 0.00858043
- good 0.00806342
- large 0.00736572
- great 0.00728838
- important 0.00710116
- other 0.0067399
- major 0.00628244
- little 0.00545736
- ...

- P(word | state 24)

- , 0.49014
- . 0.433618
- ; 0.0079789
- -- 0.00365591
- - 0.00302614
- ! 0.00235752
- : 0.001859



Example word distributions (2 of 2)

- $P(\text{word} \mid \text{state 1})$
 - unk1 0.116254
 - United+States 0.012609
 - world 0.009212
 - U.S 0.007950
 - University 0.007243
 - Internet 0.007152
 - time 0.005167
 - end 0.004928
 - unk0 0.004818
 - war 0.004260
 - country 0.003774
 - way 0.003528
 - city 0.003429
 - US 0.003269
 - Sun 0.002982
 - Earth 0.002628
 -

- $P(\text{word} \mid \text{state 3})$
 - the 0.863846
 - a 0.0131049
 - an 0.00960474
 - its 0.008541
 - our 0.00650477
 - this 0.00366675
 - unk1 0.00313899
 - your 0.00265876



Correlation between LM and IE accuracy

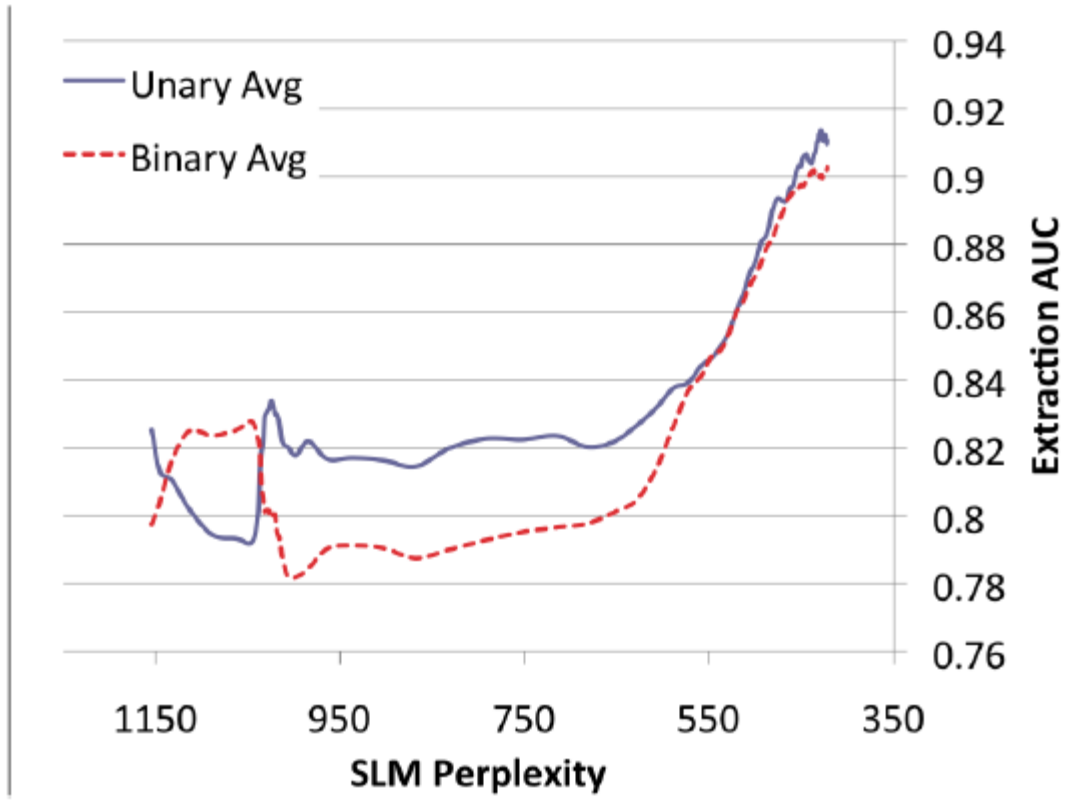
Below: correlation coefficients

As LM error decreases, IE accuracy increases

LM	Unary	Binary	Wikipedia
HMM 1-5	-.911	-.361	-.994
HMM 2-5	-.856	.120	-.930
HMM 3-5	-.823	-.683	.922
HMM 1-10	-.916	-.967	-.905
HMM 2-10	-.877	-.797	-.963
HMM 3-10	-.957	-.669	-.924
HMM 1-25	-.933	-.850	-.959
HMM 1-50	-.942	-.942	-.947
HMM 1-100	-.896	-.877	-.942
N-Gram	-.512	-.999	-

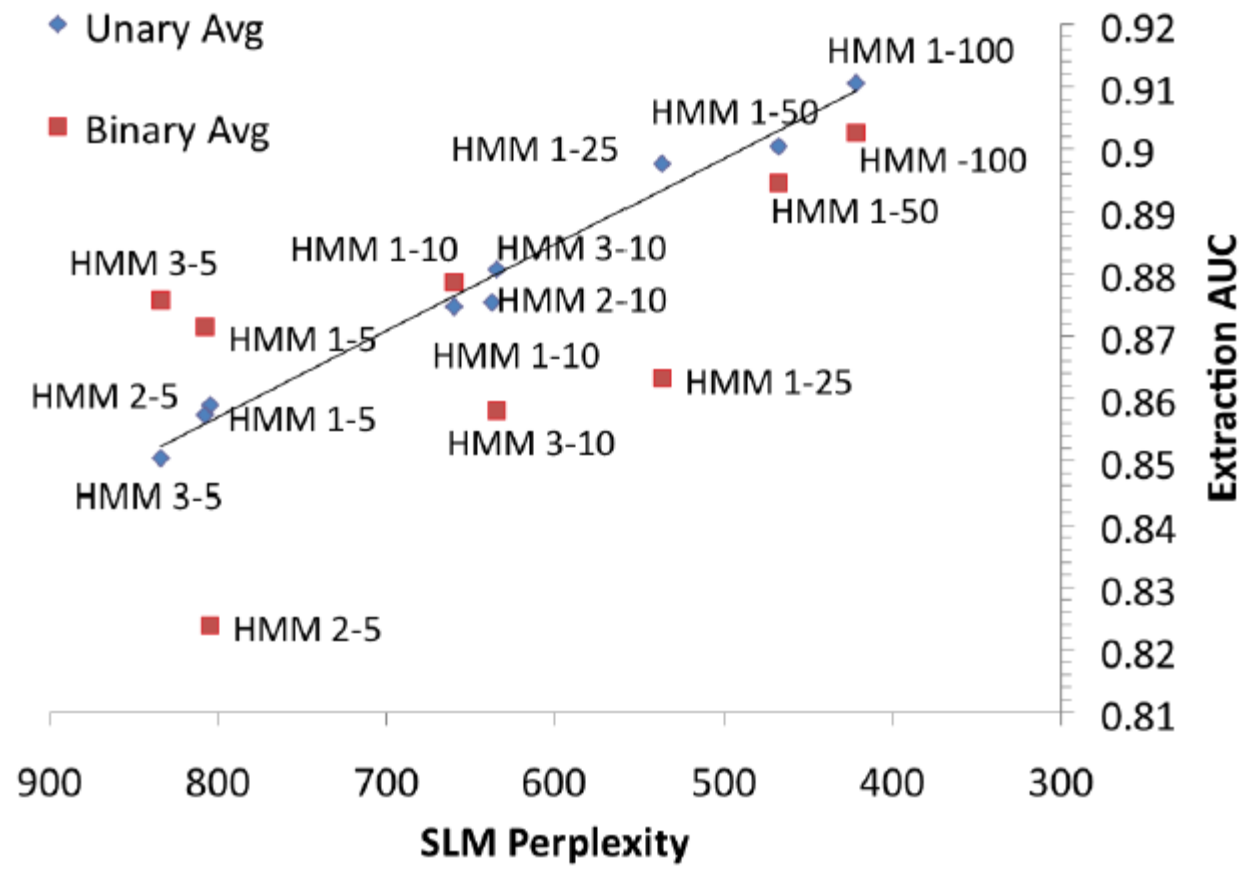


Correlation between LM and IE accuracy





Correlation between LM and IE accuracy





What this suggests

- Better HMM language models => better information extraction
- Better HMM language models => ... => human-level AI?
 - Consider: a good enough LM could do question answering, pass the Turing Test, etc.
- There are lots of paths to human-level AI, but LMs have:
 - Well-defined progress
 - Ridiculous amounts of training data



Also: active learning

- Today, people train language models by “taking what comes”
 - Larger corpora => better language models
- But corpus size limited by # of humans typing
 - What if we asked for the *most informative* sentences? (active learning)



What have we learned?

- In HMMs, general Bayes Net algorithms have simple & efficient form

1. Evaluation

GIVEN a HMM M , and a sequence \mathbf{x} ,

FIND $\text{Prob}[\mathbf{x} | M]$

Forward Algorithm and Backward Algorithm (Variable Elimination)

2. Decoding

GIVEN a HMM M , and a sequence \mathbf{x} ,

FIND the sequence π of states that maximizes $P[\mathbf{x}, \pi | M]$

Viterbi Algorithm (MAP query)

3. Learning

GIVEN A sequence \mathbf{x} ,

FIND HMM parameters $\theta = (e_i(\cdot), a_{ij})$ that maximize $P[\mathbf{x} | \theta]$

Baum-Welch/Forward-Backward algorithm (EM)



What have we learned?

- Unsupervised Learning of HMMs can power more scalable, accurate unsupervised IE