Introduction to Natural Language Processing I [Statistické metody zpracování přirozených jazyků I] (NPFL067)

http://ufal.mff.cuni.cz/courses/npfl067

prof. RNDr. Jan Hajič, Dr. / doc. RNDr. Pavel Pecina, Ph.D.
ÚFAL MFF UK
{hajic,pecina}@ufal.mff.cuni.cz
http://ufal.mff.cuni.cz/jan-hajic
http://ufal.mff.cuni.cz/~pecina/index.html

Markov Models

Review: Markov Process

- Bayes formula (chain rule):

$$P(W) = P(w_1,w_2,...,w_T) = \prod_{i=1..T} p(w_i|w_1,w_2,..,w_{i-n+1},..,w_{i-1})$$

- n-gram language models:
  - Markov process (chain) of the order n-1:

$$P(W) = P(w_1,w_2,...,w_T) = \prod_{i=1..T} p(w_i|w_{i-n+1},w_{i-n+2},..,w_{i-1})$$

approximation

Using just <u>one</u> distribution (Ex.: trigram model: $p(w_i|w_{i-2},w_{i-1})$):

Positions:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Words:	My	car	**broke**	**down**	**,**	and	within	hours	Bob	's	car	**broke**	**down**	**,**	too	.

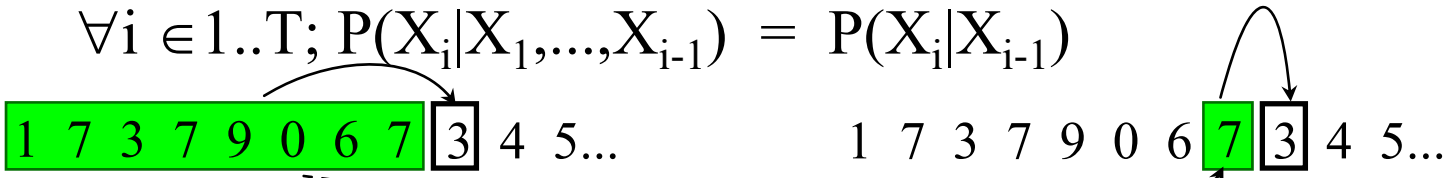
$$p(,|\textbf{broke down}) = p(w_5|w_3,w_4)) = p(w_{14}|w_{12},w_{13})$$

Markov Properties

- Generalize to any process (not just words/LM):
  - Sequence of random variables: $X = (X_1, X_2, ..., X_T)$
  - Sample space S (*states*), size N: $S = \{s_0, s_1, s_2, ..., s_N\}$

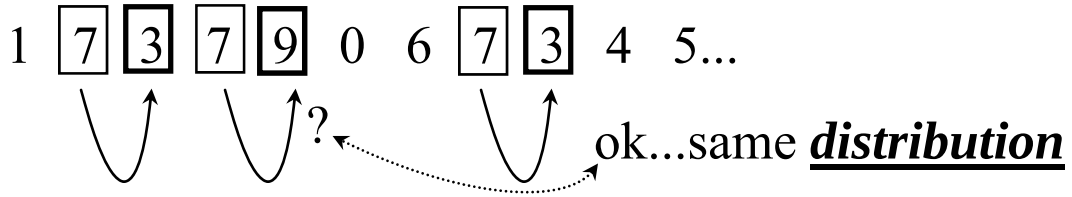
1. Limited History (Context, Horizon):

$$\forall i \in 1..T;\ P(X_i|X_1,...,X_{i-1}) = P(X_i|X_{i-1})$$



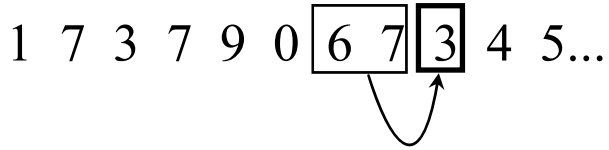
2. Time invariance (M.C. is stationary, homogeneous)

$$\forall i \in 1..T,\ \forall y,x \in S;\ P(X_i=y|X_{i-1}=x) = p(y|x)$$



Long History Possible

- What if we want trigrams:

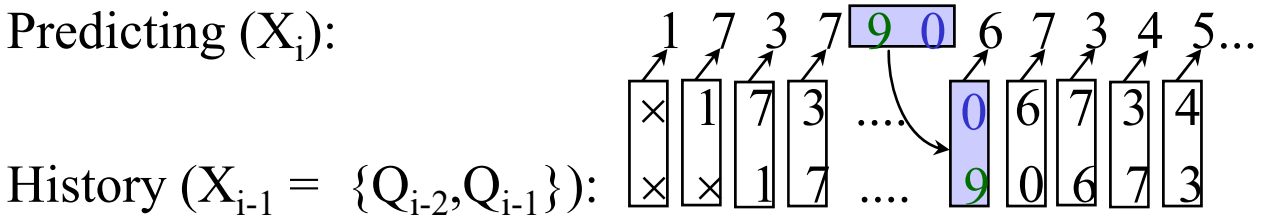


- Formally, use transformation:

Define new variables Q_i, such that $X_i = \{Q_{i-1},Q_i\}$:

Then

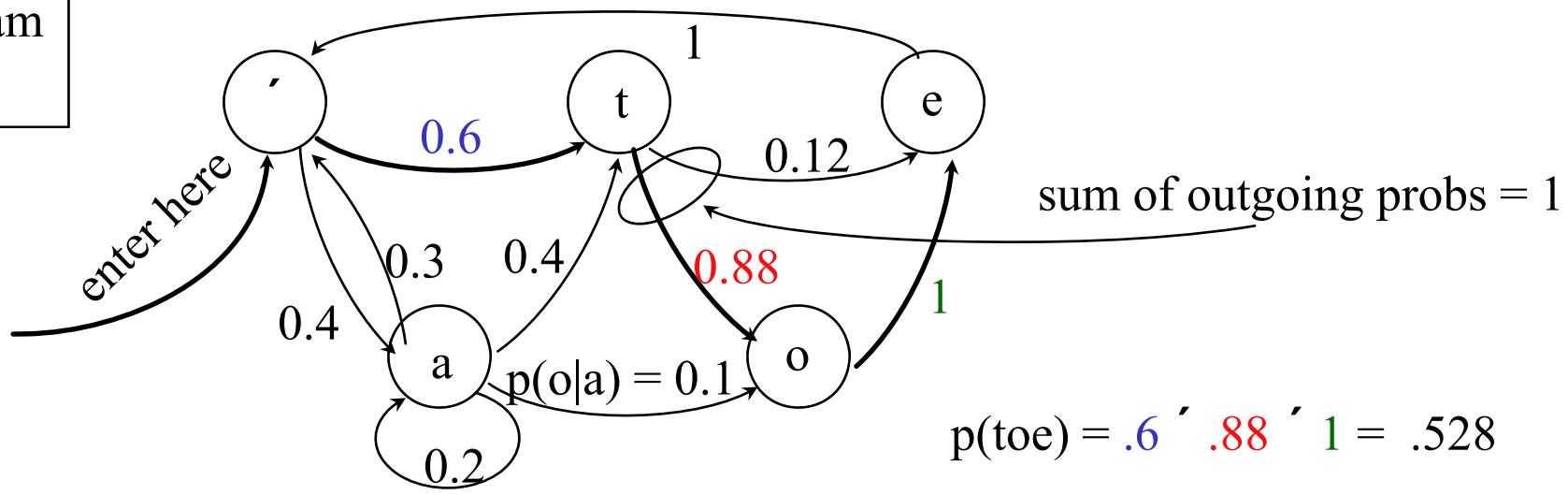
$$P(X_i|X_{i-1}) = P(Q_{i-1},Q_i|Q_{i-2},Q_{i-1}) = P(Q_i|Q_{i-2},Q_{i-1})$$



Graph Representation: State Diagram

- S = $\{s_0,s_1,s_2,...,s_N\}$: states
- Distribution $P(X_i|X_{i-1})$:
  - **transitions (as arcs) with probabilities attached to them:**

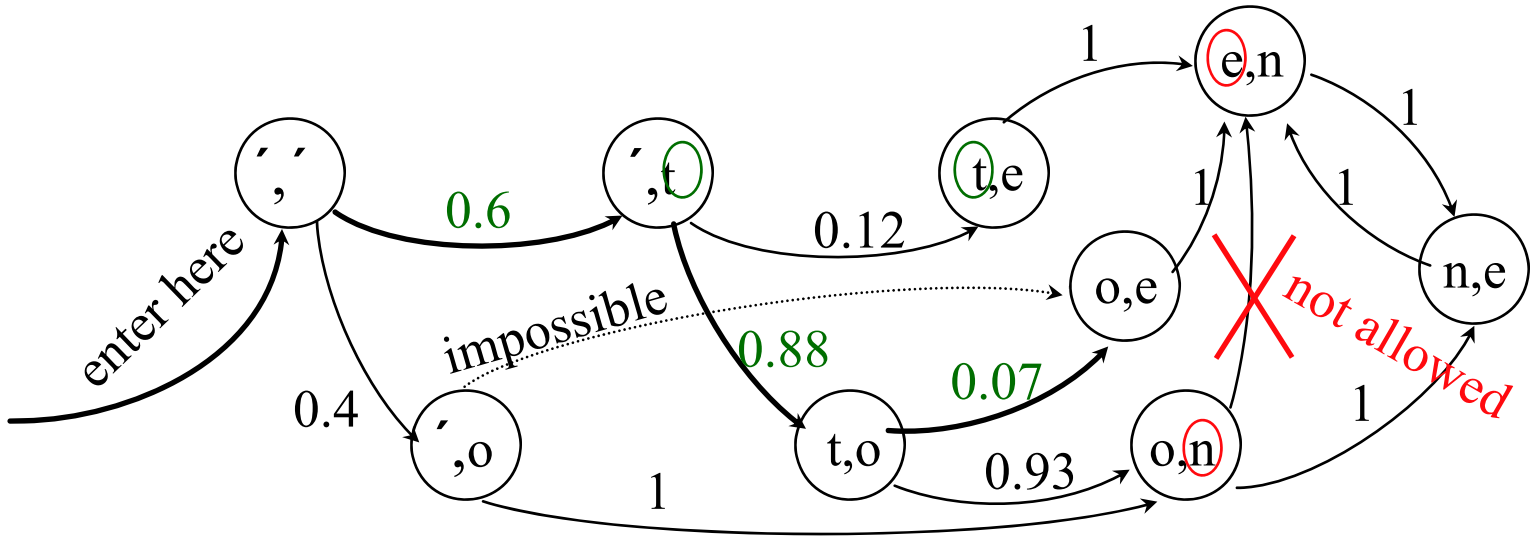
Bigram case:



p(toe) = .6 ´ .88 ´ 1 = .528

The Trigram Case

- $S = \{s_0,s_1,s_2,...,s_N\}$: states: pairs $s_i = (x,y)$
- Distribution $P(X_i|X_{i-1})$: (r.v. X: generates pairs s_i)

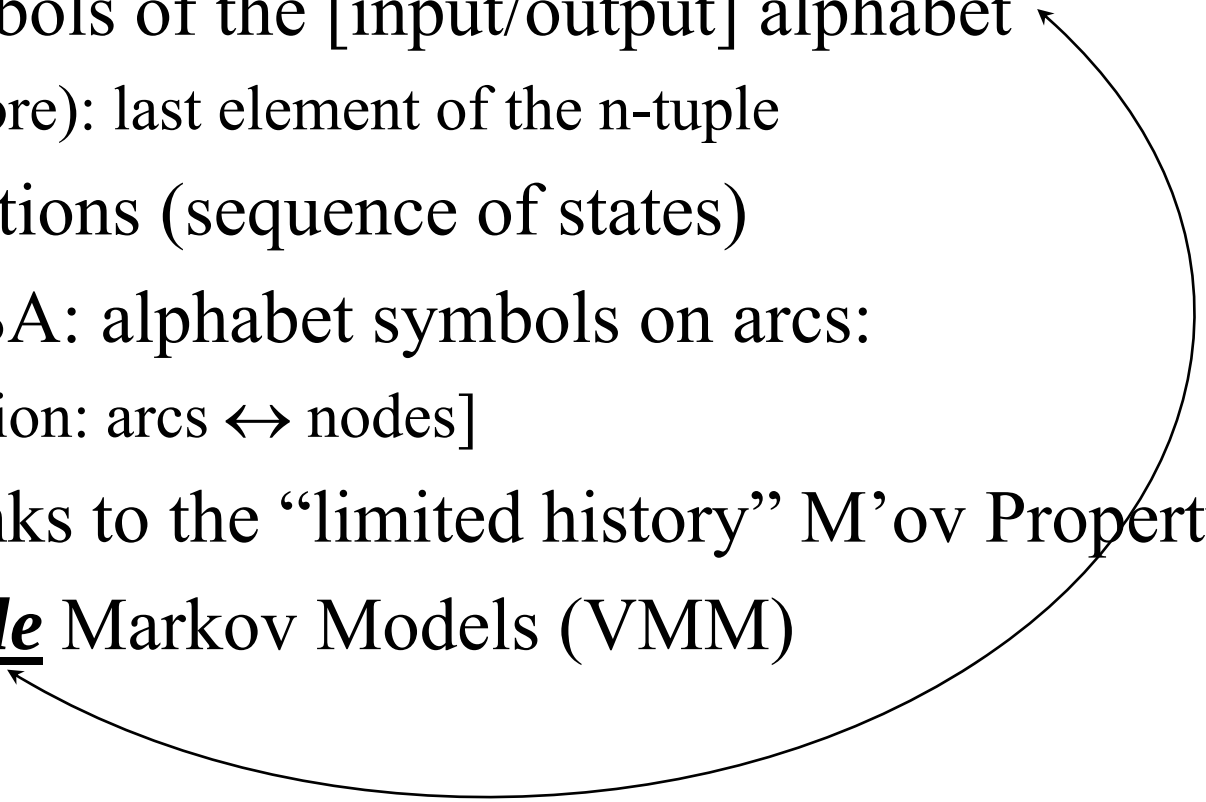


p(toe) = .6 ´ .88 ´ .07 ≅ .037

p(one) = ?

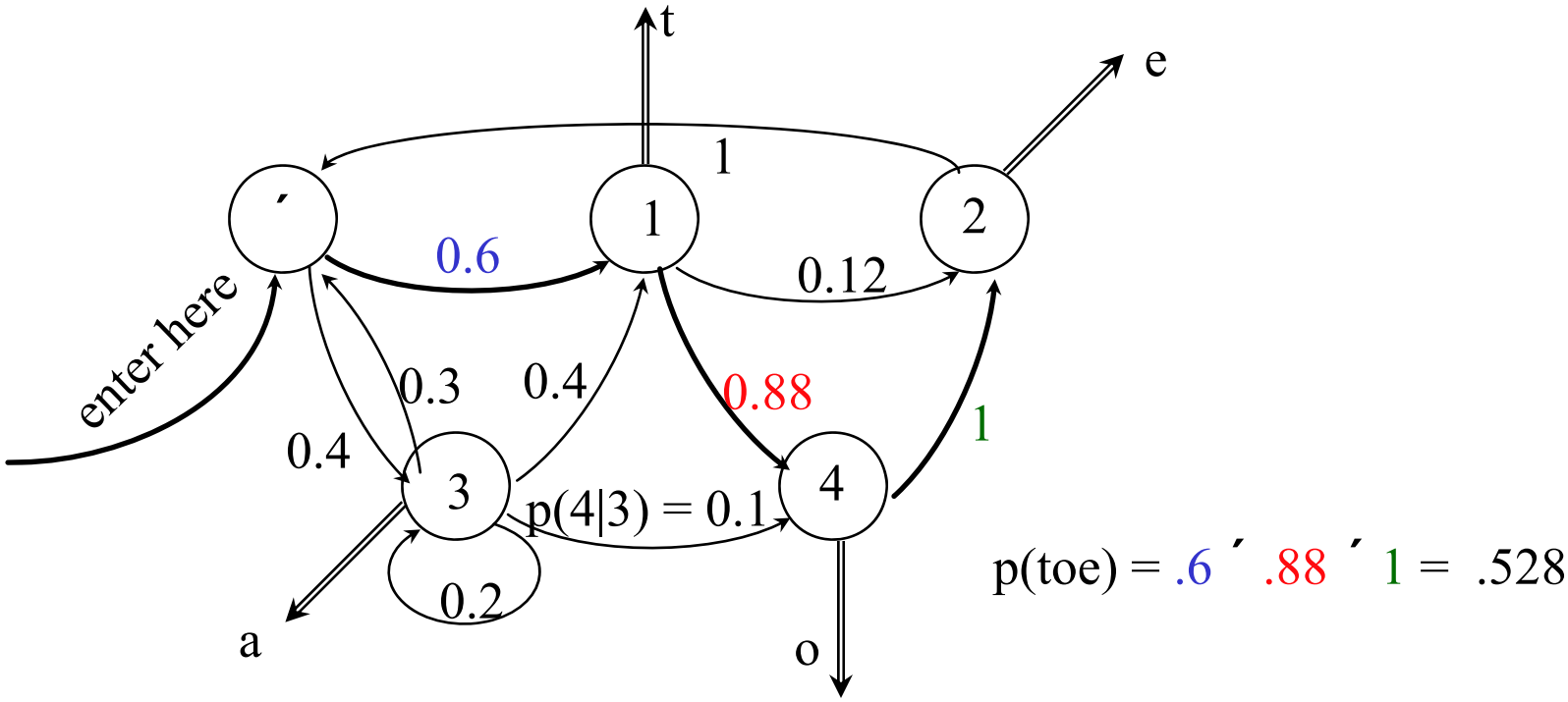
Finite State Automaton

- States ~ symbols of the [input/output] alphabet
  - pairs (or more): last element of the n-tuple
- Arcs ~ transitions (sequence of states)
- [Classical FSA: alphabet symbols on arcs:
  - transformation: arcs $\leftrightarrow$ nodes]
- Possible thanks to the "limited history" M'ov Property
- So far: ***Visible*** Markov Models (VMM)



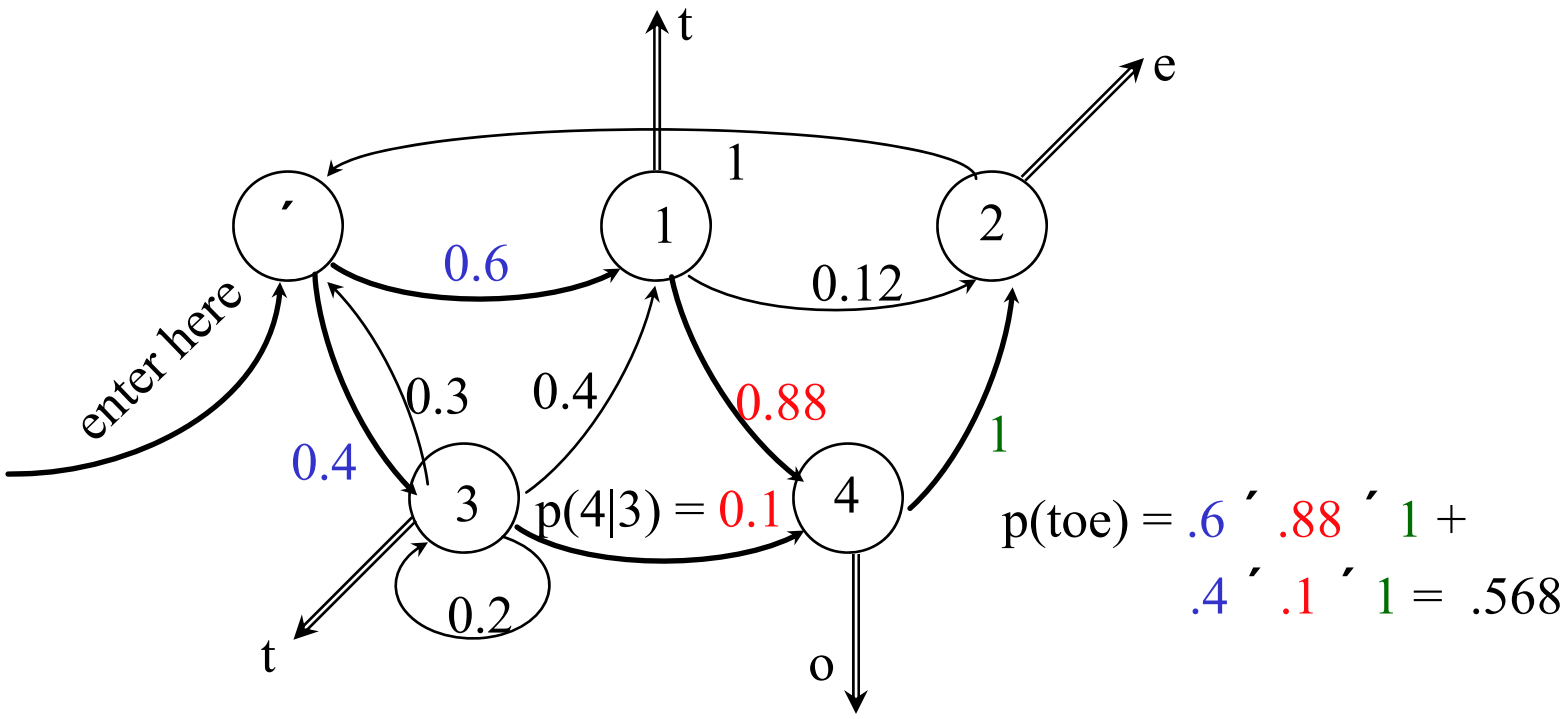
Hidden Markov Models

- The simplest HMM: states generate [observable] output (using the "data" alphabet) but remain "invisible":



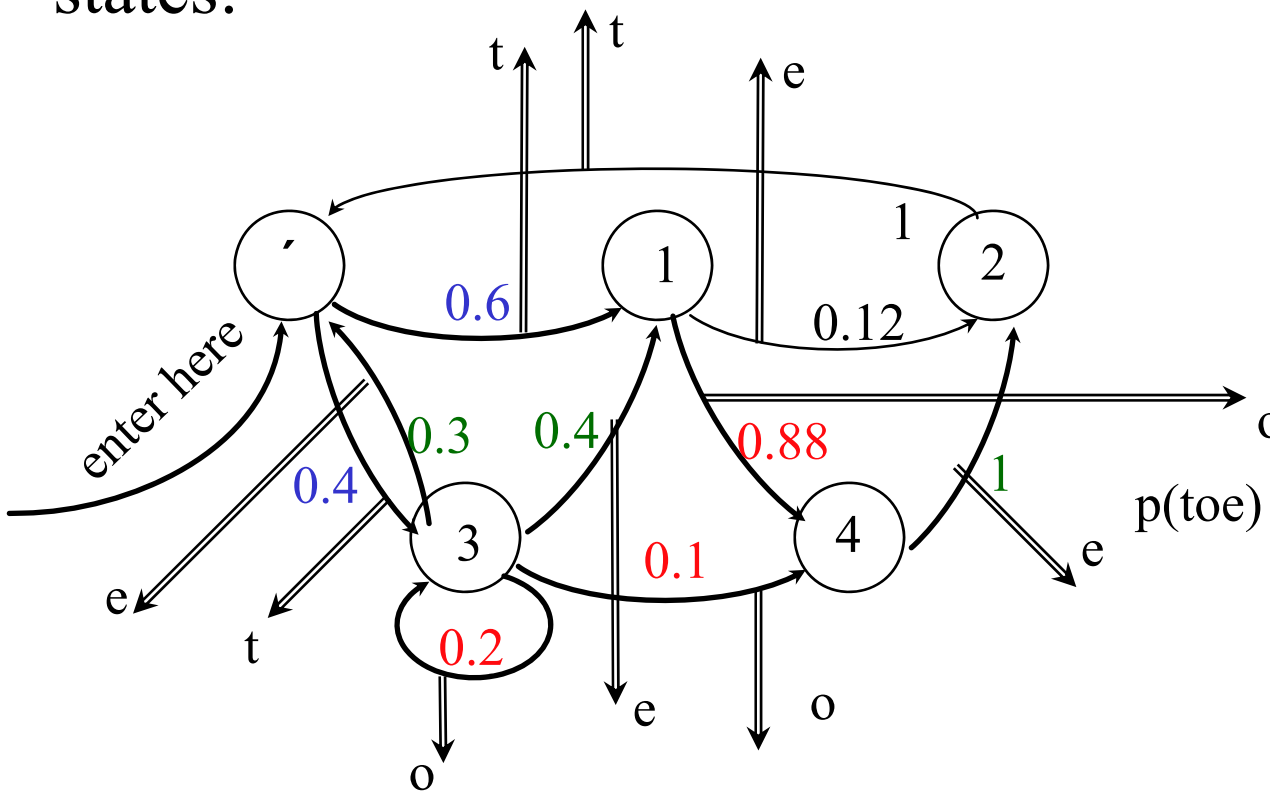
Added Flexibility

- So far, no change; but different states may generate the same output (why not?):



Output from Arcs...

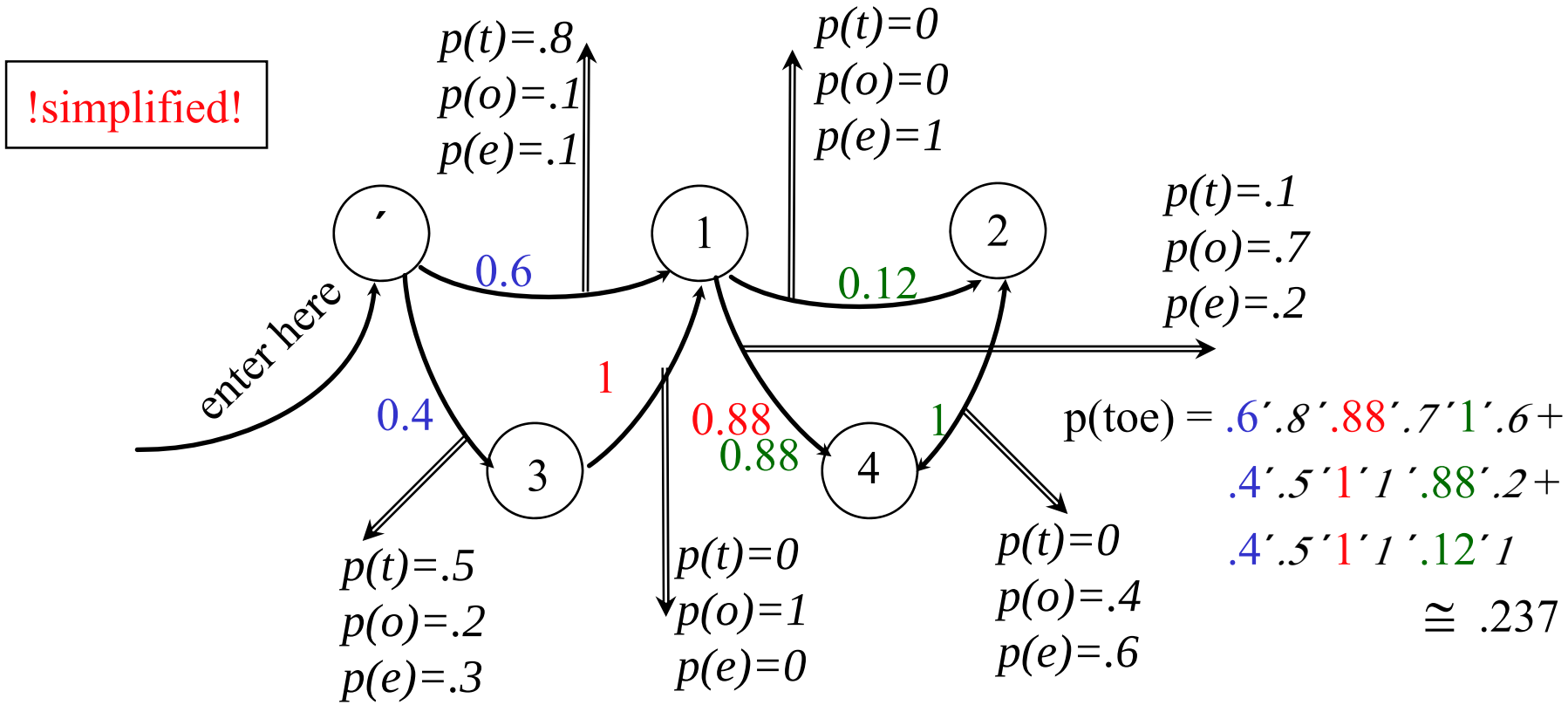
- Added flexibility: Generate output from arcs, not states:



p(toe) = .6 ´ .88 ´ 1 +
.4 ´ .1 ´ 1 +
.4 ´ .2 ´ .3 +
.4 ´ .2 ´ .4 = .624

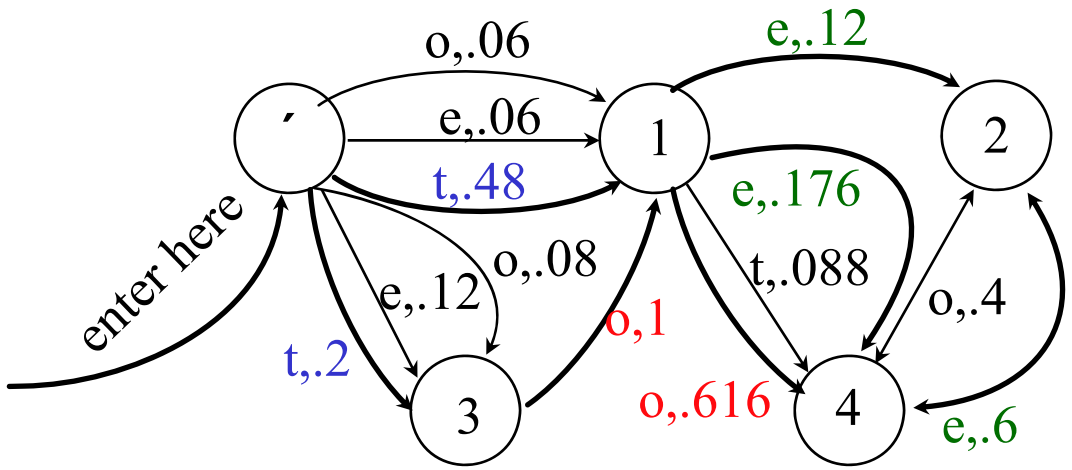
... and Finally, Add Output Probabilities

- Maximum flexibility: [Unigram] distribution (sample space: output alphabet) at each output arc:



Slightly Different View

- Allow for multiple arcs from $s_i \rightarrow s_j$, mark them by output symbols, get rid of output distributions:



p(toe) = .48´.616´.6+
.2´1´.176 +
.2´1´.12 ≅ .237

In the future, we will use the view more convenient for the problem at hand.

Formalization

- HMM (the most general case):
  - five-tuple (S, s_0, Y, P_S, P_Y), where:
    - **$S = \{s_0, s_1, s_2, ..., s_T\}$ is the set of states, s_0 is the initial state,**
    - **$Y = \{y_1, y_2, ..., y_V\}$ is the output alphabet,**
    - **$P_S(s_j|s_i)$ is the set of prob. distributions of transitions,**
      - size of P_S: $|S|^2$.
    - **$P_Y(y_k|s_i,s_j)$ is the set of output (emission) probability distributions.**
      - size of P_Y: $|S|^2 \times |Y|$
- Example:
  - $S = \{x, 1, 2, 3, 4\}$, $s_0 = x$
  - $Y = \{ t, o, e \}$

Formalization - Example

- Example (for graph, see foils 11,12):
  - $S = \{x, 1, 2, 3, 4\}$, $s_0 = x$
  - $Y = \{ e, o, t \}$
  - P_S:

	x	1	2	3	4
x	0	.6	0	.4	0
1	0	0	.12	0	.88
2	0	0	0	0	1
3	0	1	0	0	0
4	0	0	1	0	0

$\Sigma = 1$

P_Y:

		e	x	1	2	3	4
		o	x	1	2	3	4
t	x	1	2	3	4		
x		.8		.5		.7	.2
1			0		.1		
2					0		
3		0					
4			0				

$\Sigma = 1$

Using the HMM

- The generation algorithm (of limited value :-)):
  1. **Start in $s = s_0$.**
  2. **Move from s to s' with probability $P_S(s'|s)$.**
  3. **Output (emit) symbol y_k with probability $P_S(y_k|s,s')$.**
  4. **Repeat from step 2 (until somebody says enough).**
- More interesting usage:
  - Given an output sequence $Y = \{y_1,y_2,...,y_k\}$, compute its probability.
  - Given an output sequence $Y = \{y_1,y_2,...,y_k\}$, compute the most likely sequence of states which has generated it.
  - ...plus variations: e.g., n best state sequences