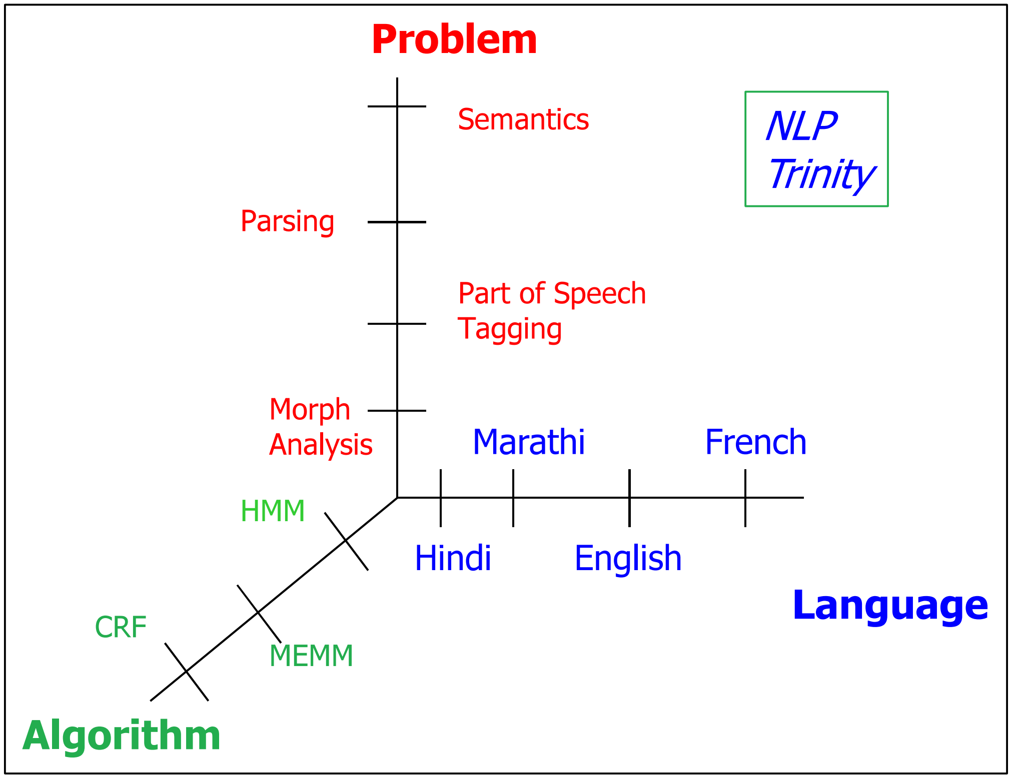
CS626: NLP, Speech and the Web

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

Lecture 6, 7, 8, 9: Viterbi; forward and backward; Baum Welch; IL POS tags

6th, 8th, 9th, 13th August, 2012

HMM


Problem
Semantics
NLP Trinity
Parsing
Part of Speech Tagging
Morph Analysis
Marathi
French
HMM
Hindi
English
CRF
MEMM
Language
Algorithm

HMM Definition

- Set of states: S where |S|=N
- Start state S_0 /*P(S_0)=1*/
- Output Alphabet: O where |O|=M
- Transition Probabilities: A= {a_{ij}} /*state i to state j*/
- Emission Probabilities : B= {$b_j(o_k)$} /*prob. of emitting or absorbing o_k from state j*/
- Initial State Probabilities: $\Pi=\{p_1,p_2,p_3,\ldots p_N\}$
- Each $p_i=P(o_0=\varepsilon,S_i|S_0)$

Markov Processes

- Properties
  - Limited Horizon: Given previous *t* states, a state *i*, is independent of preceding *0* to *t-k+1* states.
    - $P(X_t=i|X_{t-1}, X_{t-2},\ldots X_0) = P(X_t=i|X_{t-1}, X_{t-2}\ldots X_{t-k})$
    - Order *k* Markov process
  - Time invariance: (shown for *k=1*)
    - $P(X_t=i|X_{t-1}=j) = P(X_1=i|X_0=j) \ldots = P(X_n=i|X_{n-1}=j)$

Three basic problems (contd.)

- Problem 1: Likelihood of a sequence
  - Forward Procedure
  - Backward Procedure
- Problem 2: Best state sequence
  - Viterbi Algorithm
- Problem 3: Re-estimation
  - Baum-Welch (Forward-Backward Algorithm)

Probabilistic Inference

- O: Observation Sequence
- S: State Sequence

- Given O find S^* where $S^* = \arg\max_{S} p(S/O)$ called Probabilistic Inference

- Infer "Hidden" from "Observed"
- How is this inference different from logical inference based on propositional or predicate calculus?

Essentials of Hidden Markov Model

1. Markov + Naive Bayes

2. Uses both transition and observation probability

$$p(S_k \xrightarrow{O_k} S_{k+1}) = p(O_k / S_k)\, p(S_{k+1} / S_k)$$

3. Effectively makes Hidden Markov Model a Finite State Machine (FSM) with probability

Probability of Observation Sequence

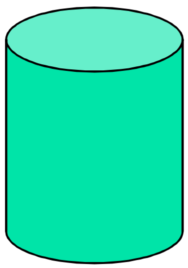
$$p(O) = \sum_S p(O, S)$$

$$= \sum_S p(S)\, p(O / S)$$

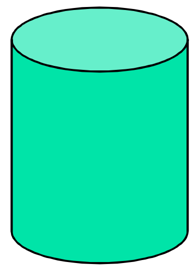
- Without any restriction,
  - Search space size= $|S|^{|O|}$

Continuing with the Urn example

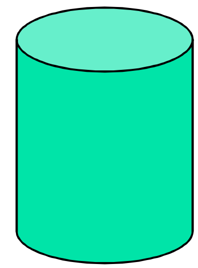
Colored Ball choosing



Urn 1
# of Red = 30
# of Green = 50
# of Blue = 20



Urn 2
# of Red = 10
# of Green = 40
# of Blue = 50



Urn 3
# of Red =60
# of Green =10
# of Blue = 30

Example (contd.)

Given :

Transition Probability

	U_1	U_2	U_3
U_1	0.1	0.4	0.5
U_2	0.6	0.2	0.2
U_3	0.3	0.4	0.3

and

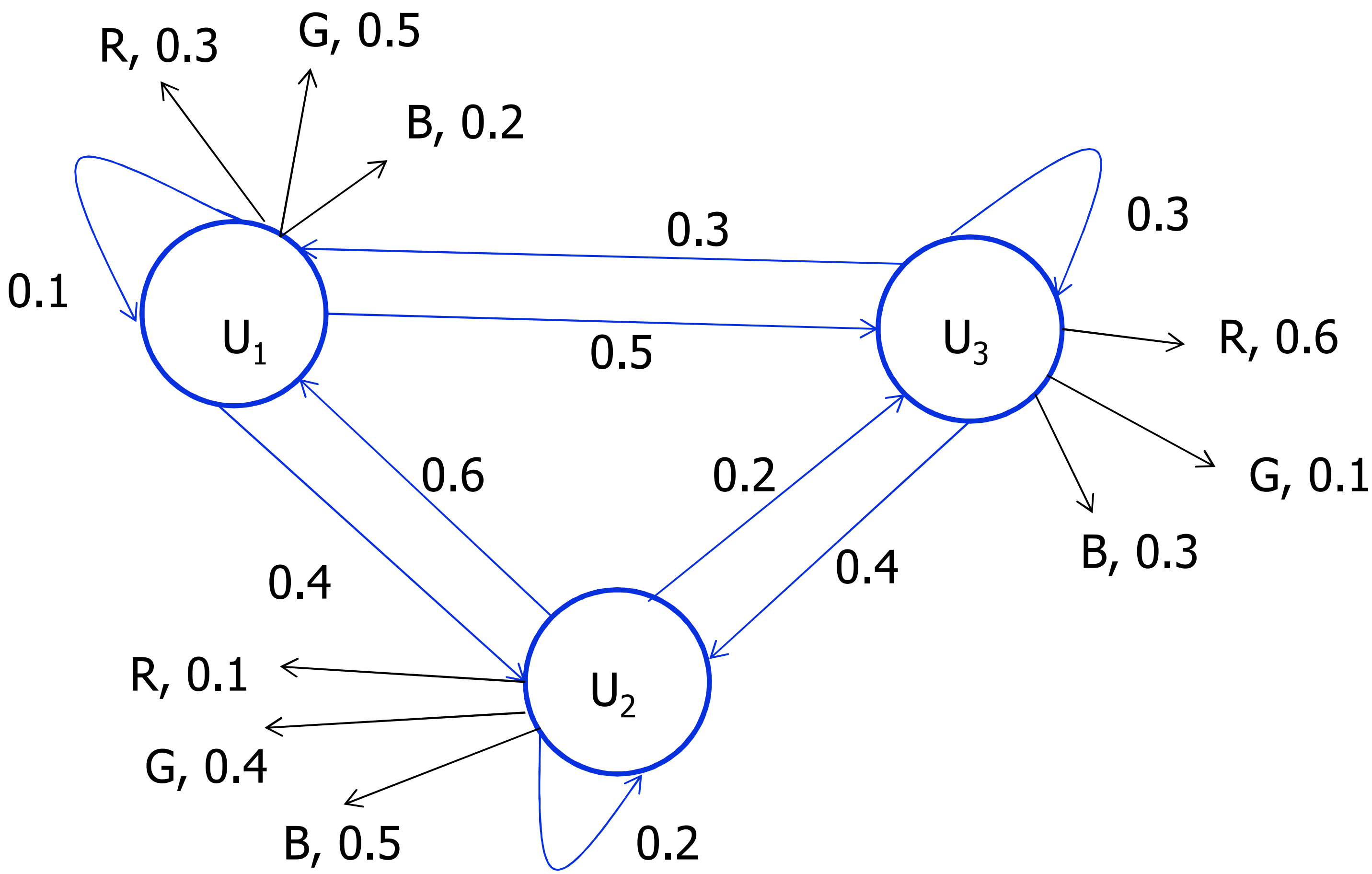
Observation/output Probability

	R	G	B
U_1	0.3	0.5	0.2
U_2	0.1	0.4	0.5
U_3	0.6	0.1	0.3

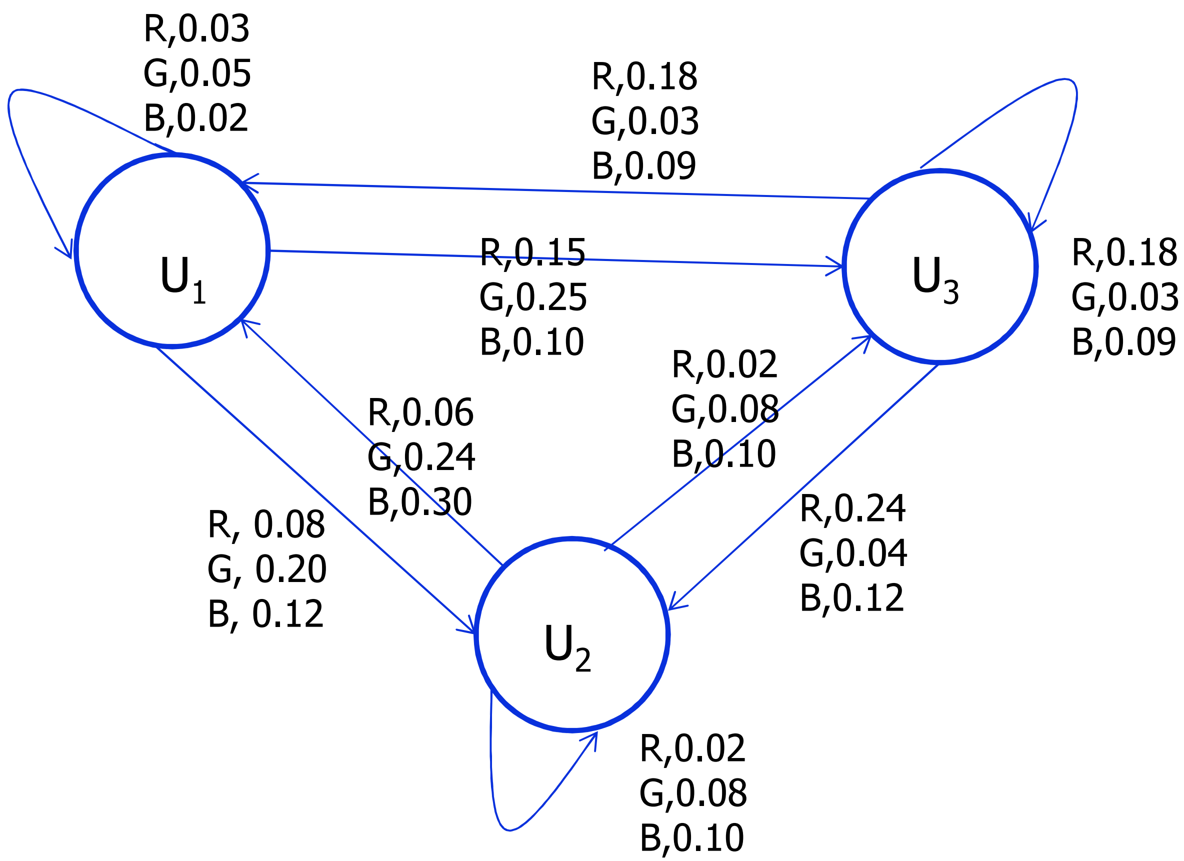
Observation : RRGGBRGR

What is the corresponding state sequence ?

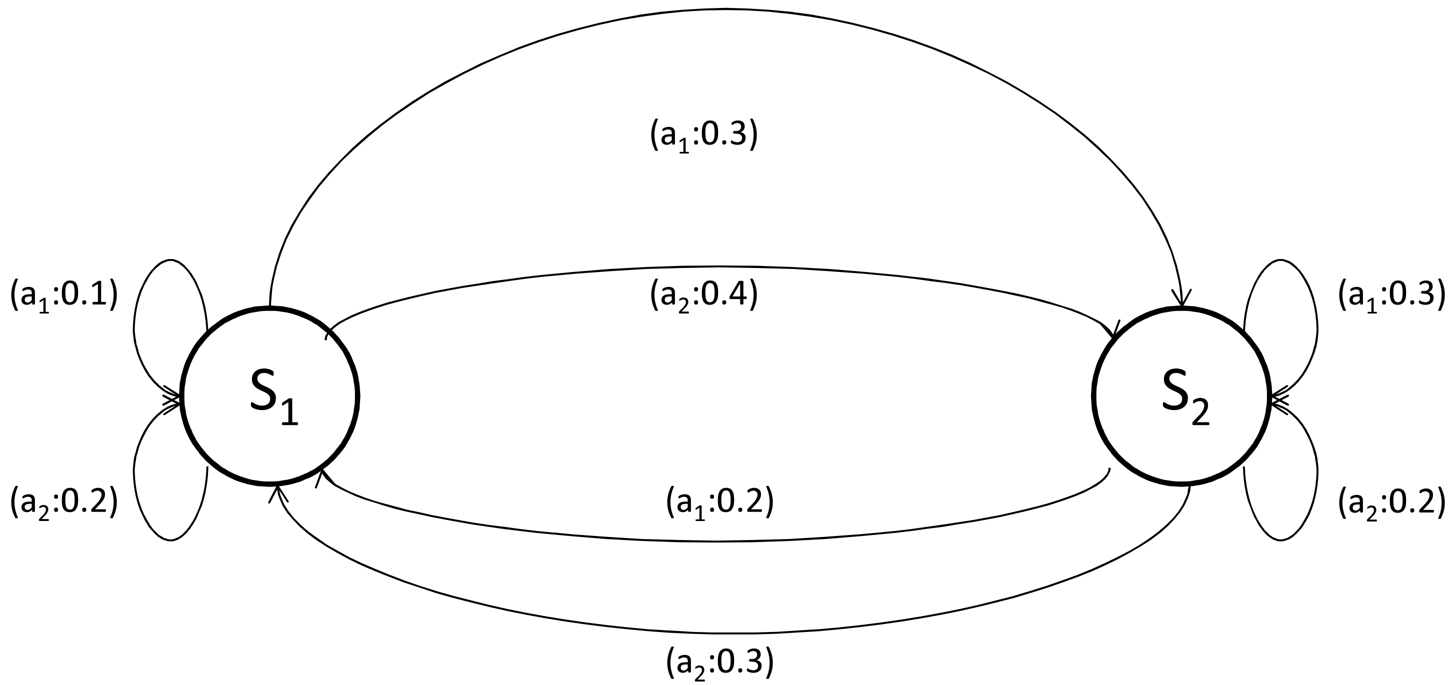
Diagrammatic representation (1/2)



Diagrammatic representation (2/2)

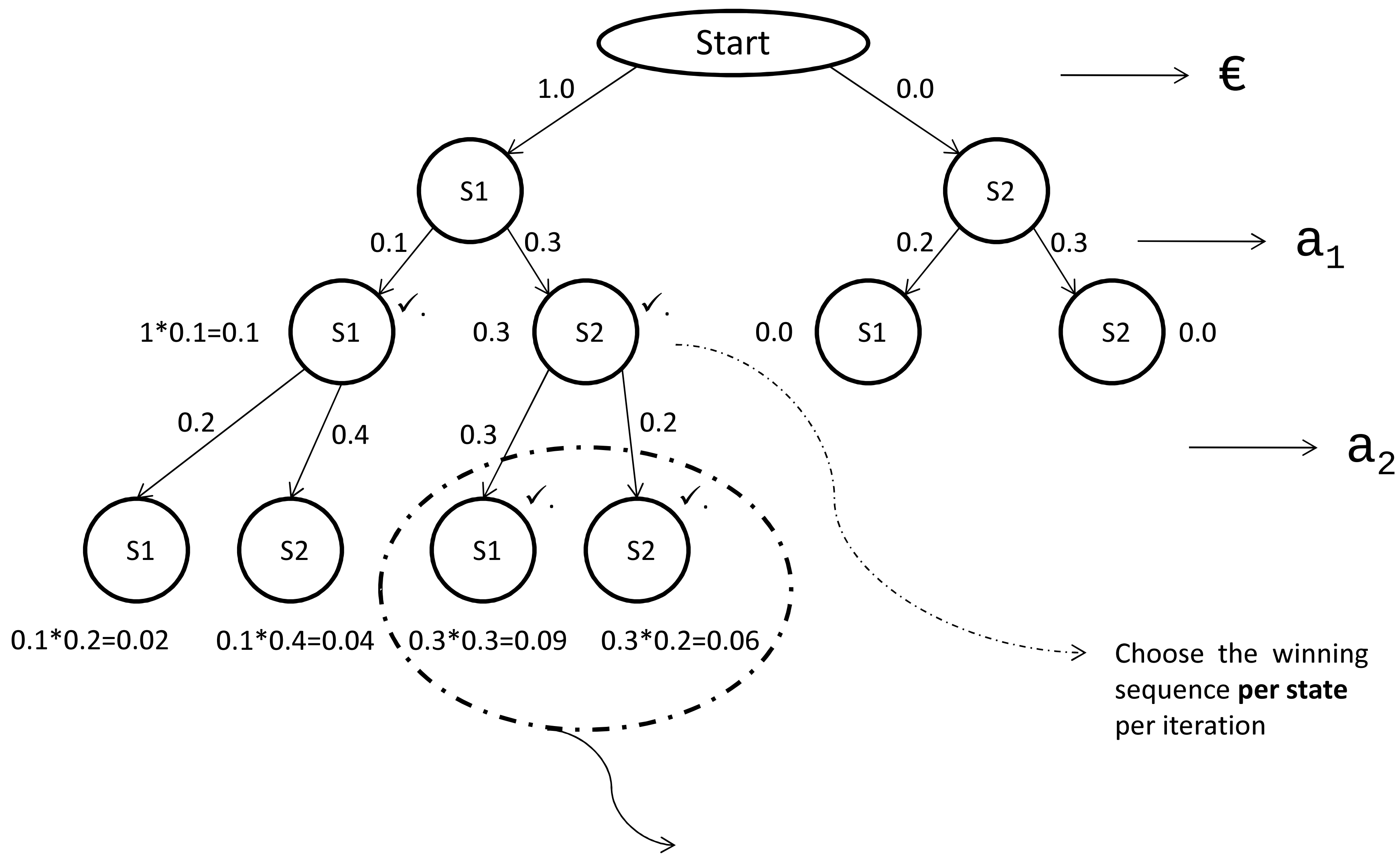


Probabilistic FSM

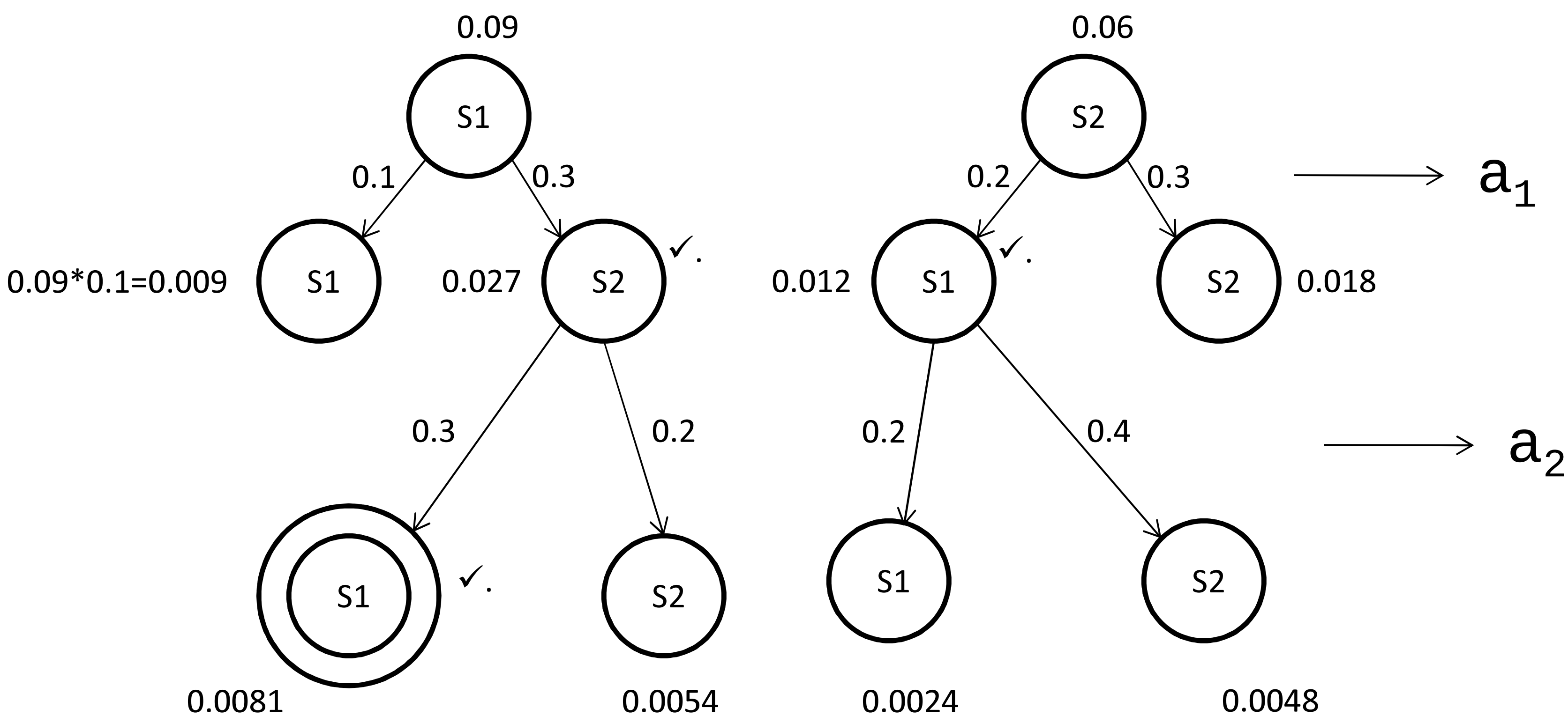


The question here is:
"what is the most likely state sequence given the output sequence seen"

Developing the tree

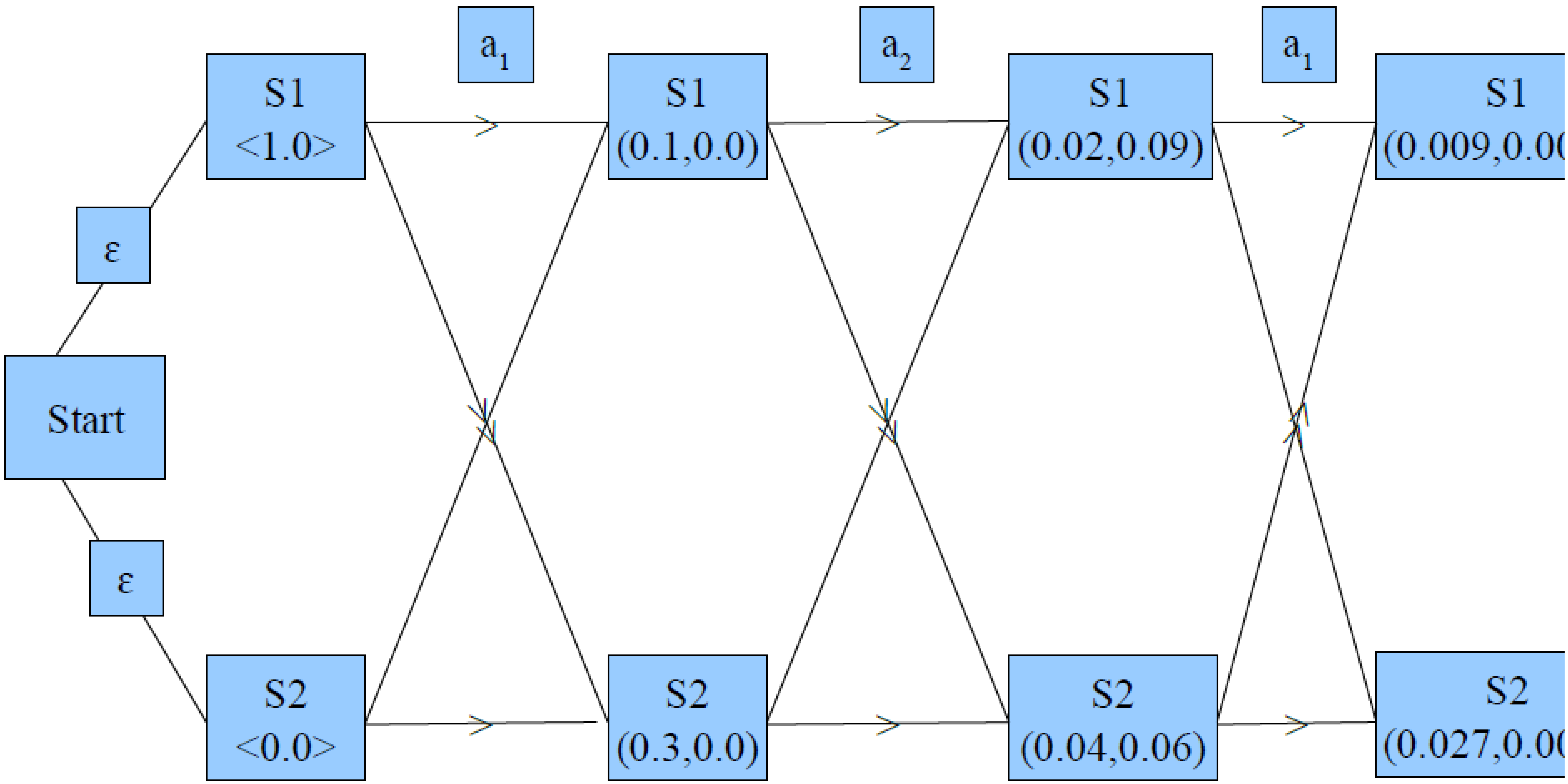


Tree structure contd…



The problem being addressed by this tree is $S^* = \arg\max_{S} P(S \mid a_1 - a_2 - a_1 - a_2, \mu)$

a1-a2-a1-a2 is the output sequence and μ the model or the machine



Start
ε
ε
S1
<1.0>
S2
<0.0>
a1
S1
(0.1,0.0)
S2
(0.3,0.0)
a2
S1
(0.02,0.09)
S2
(0.04,0.06)
a1
S1
(0.009,0.0
S2
(0.027,0.0

Tabular representation of the tree

Latest symbol observed / Ending state	€	a_1	a_2	a_1	a_2
S_1	1.0	(1.0*0.1,0.0*0.2)=(**0.1**,0.0)	(0.02, **0.09**)	(0.009, **0.012**)	(0.0024, **0.0081**)
S_2	0.0	(1.0*0.3,0.0*0.3)=(**0.3**,0.0)	(0.04,**0.06**)	(**0.027**,0.018)	(0.0048,0.0054)

Note: Every cell records the winning probability ending in that state

Final winner

The bold faced values in each cell shows the sequence probability ending in that state. Going backward from final winner sequence which ends in state S_2 (indicated By the 2nd tuple), we recover the sequence.

Algorithm
(following James Alan, Natural Language Understanding (2nd edition), Benjamin Cummins (pub.), 1995

Given:

1. The HMM, which means:
   a. Start State: S_1
   b. Alphabet: $A = \{a_1, a_2, \ldots a_p\}$
   c. Set of States: $S = \{S_1, S_2, \ldots S_N\}$
   d. Transition probability $P(S_i \xrightarrow{a_k} S_j) \quad \forall_{i,j,k}$
      which is equal to $P(S_j, a_k \mid S_i)$
2. The output string $a_1 a_2 \ldots a_M$

To find:

The most likely sequence $C_1 C_2 \ldots C_M$ which produces the given output sequence, *i.e.*, $C_1 C_2 \ldots C_M = \text{argmax}_C(P(C \mid a_1 a_2 \ldots a_M)$

Algorithm contd…

Data Structure:

1. A N*M array called SEQSCORE to maintain the winner sequence always (N=#states, M=length of o/p sequence)
2. Another N*M array called BACKPTR to recover the path.

Three distinct steps in the Viterbi implementation

1. Initialization
2. Iteration
3. Sequence Identification

1. Initialization

SEQSCORE(1,1)=1.0

BACKPTR(1,1)=0

For(i=2 to N) do

SEQSCORE(i,1)=0.0

[expressing the fact that first state is S_1]

2. Iteration

For(t=2 to T) do

For(i=1 to N) do

SEQSCORE(i,t) = Max$_{(j=1,N)}$

$$[SEQSCORE(j,(t-1)) * P(Sj \xrightarrow{a_k} Si)]$$

BACKPTR(I,t) = index *j* that gives the MAX above

3. Seq. Identification

C(M) = i that maximizes SEQSCORE(i,M)

For i from (M-1) to 1 do

C(i) = BACKPTR[C(i+1),(i+1)]

Optimizations possible:

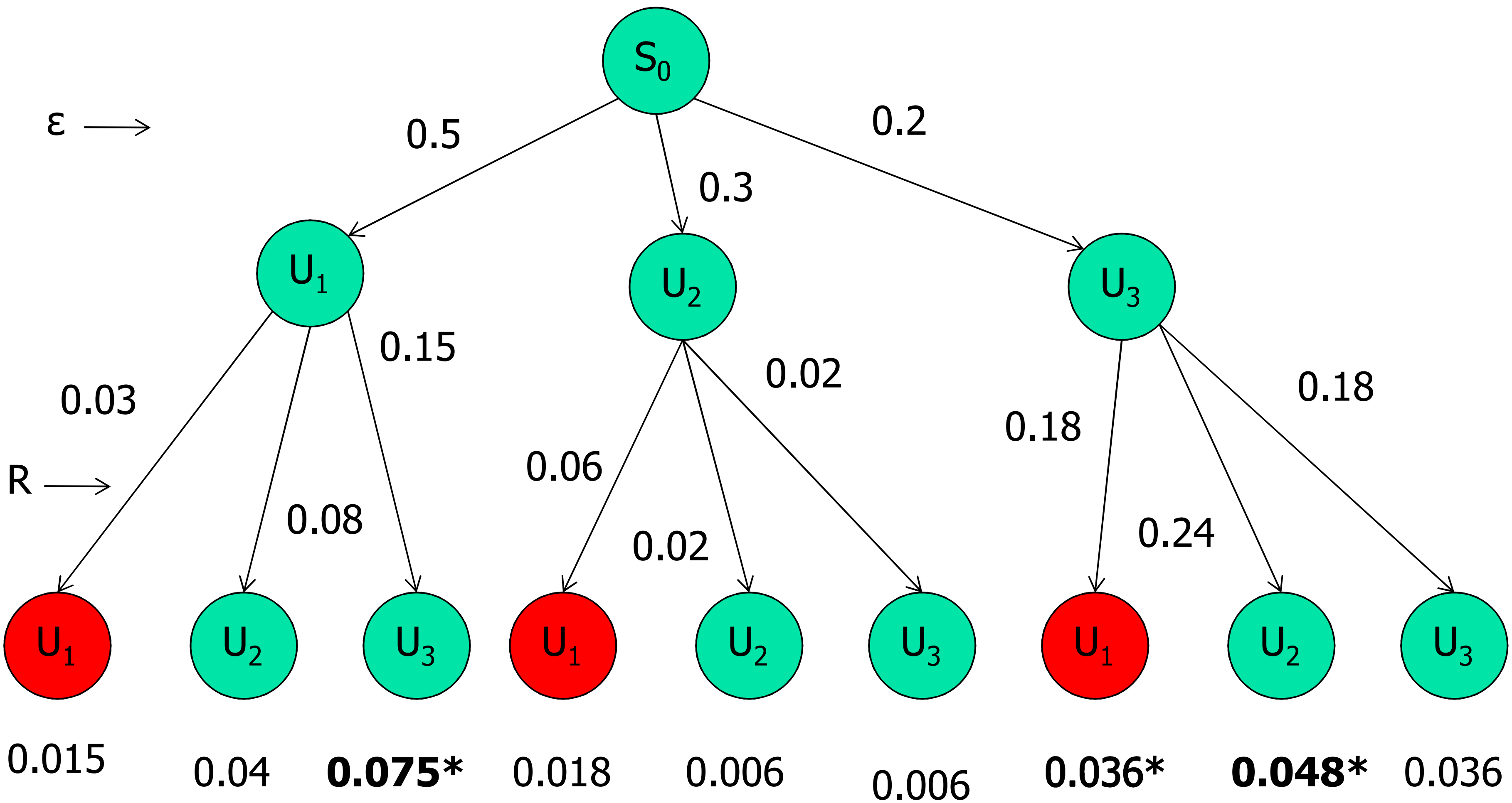
1. BACKPTR can be 1*M
2. SEQSCORE can be M*2

Homework:- Compare this with A*, Beam Search [Homework]

Reason for this comparison:

Both of them work for finding and recovering sequence

Viterbi Algorithm for the Urn problem (first two symbols)



: winner sequences

Markov process of order>1 (say 2)

	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	
Obs:	ε	R	R	G	G	B	R	G	R	
State:	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9

Same theory works

$P(S).P(O|S)$

$= P(O_0|S_0).P(S_1|S_0).$
$[P(O_1|S_1). \quad P(S_2|S_1S_0)].$
$[P(O_2|S_2). \quad P(S_3|S_2S_1)].$
$[P(O_3|S_3).P(S_4|S_3S_2)].$
$[P(O_4|S_4).P(S_5|S_4S_3)].$
$[P(O_5|S_5).P(S_6|S_5S_4)].$
$[P(O_6|S_6).P(S_7|S_6S_5)].$
$[P(O_7|S_7).P(S_8|S_7S_6)].$
$[P(O_8|S_8).P(S_9|S_8S_7)].$

We introduce the states S_0 and S_9 as initial and final states respectively.

After S_8 the next state is S_9 with probability 1, i.e., $P(S_9|S_8S_7)=1$

O_0 is ε-transition

Adjustments

- Transition probability table will have tuples on rows and states on columns
- Output probability table will remain the same
- In the Viterbi tree, the Markov process will take effect from the 3rd input symbol (εRR)
- There will be 27 leaves, out of which **only 9 will remain**
- Sequences ending in **same tuples** will be compared
  - Instead of U1, U2 and U3
  - U_1U_1, U_1U_2, U_1U_3, U_2U_1, U_2U_2,U_2U_3, U_3U_1,U_3U_2,U_3U_3

Forward and Backward Probability Calculation

Forward probability $F(k,i)$

- Define $F(k,p)$= Probability of being in state S_i having seen $o_0 o_1 o_2 \ldots o_k$
- $F(k,i)=P(o_0 o_1 o_2 \ldots o_k , S_p)$
- With m as the length of the observed sequence
- $P(observed\ sequence)=P(o_0 o_1 o_2 .. o_m)$

$$=\Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_m , S_p)$$

$$=\Sigma_{p=0,N} F(m , p)$$

Forward probability (contd.)

$F(k\,,\,q)$

$= P(o_0o_1o_2..o_k\,,\,S_q)$

$= P(o_0o_1o_2..o_k\,,\,S_q)$

$= P(o_0o_1o_2..o_{k-1}\,,\,o_k\,,S_q)$

$= \Sigma_{p=0,N}\, P(o_0o_1o_2..o_{k-1}\,,\,S_p\,,\,o_k\,,S_q)$

$= \Sigma_{p=0,N}\, P(o_0o_1o_2..o_{k-1}\,,\,S_p\,).$

$\qquad P(o_m\,,S_q|o_0o_1o_2..o_{k-1}\,,\,S_p)$

$= \Sigma_{p=0,N}\, F(k\text{-}1,p).\ P(o_k\,,S_q|S_p)$

$= \Sigma_{p=0,N}\, F(k\text{-}1,p).\ P(S_p \xrightarrow{o_k} S_q)$

	O_0	O_1	O_2	O_3 …		O_k		O_{k+1}	…	O_{m-1}	O_m	
S_0 $\longrightarrow$	S_1	S_2	S_3		… S_p $\longrightarrow$		S_q		…		S_m	S_{final}

Backward probability $B(k,i)$

- Define $B(k,i)$= Probability of seeing $o_k o_{k+1} o_{k+2} \ldots o_m$ given that the state was S_i
- $B(k,i)=P(o_k o_{k+1} o_{k+2} \ldots o_m \mid S_i)$
- With m as the length of the observed sequence
- $P(observed\ sequence)=P(o_0 o_1 o_2 .. o_m)$

$$= P(o_0 o_1 o_2 .. o_m / S_0)$$

$$=B(0,0)$$

Backward probability (contd.)

$B(k, p)$

$= P(o_k o_{k+1} o_{k+2} \ldots o_m \mid S_p)$

$= P(o_{k+1} o_{k+2} \ldots o_m, o_k / S_p)$

$= \Sigma_{q=0,N} P(o_{k+1} o_{k+2} \ldots o_m, o_k, S_q / S_p)$

$= \Sigma_{q=0,N} P(o_k, S_q / S_p)$
$P(o_{k+1} o_{k+2} \ldots o_m / o_k, S_q, S_p)$

$= \Sigma_{q=0,N} P(o_{k+1} o_{k+2} \ldots o_m / S_q). P(o_k, S_q / S_p)$

$= \Sigma_{q=0,N} B(k+1,q). P(S_p \xrightarrow{o_k} S_q)$

$O_0 \quad O_1 \quad O_2 \quad O_3 \ldots \quad O_k \quad O_{k+1} \quad \ldots \quad O_{m-1} \quad O_m$

$S_0 \rightarrow S_1 \quad S_2 \quad S_3 \quad \ldots \quad S_p \rightarrow S_q \quad \ldots \quad S_m \quad S_{final}$

Assignment 2

- Prove that a language model based on POS tagged text is better than one developed from raw text

$$LM_{pos} > Lm_{raw}$$

- Choose a suitable NLP task to compare the models
  - Eg: next word prediction

Example

- Sentence: People laugh.
  - People and laugh can both be used as noun and verb
  - N V : P(“people laugh”, N V) = P(N).P(V|N).P(people|N).P(laugh|V)
  - N N : P(“people laugh”, N N) = P(N).P(N|N).P(people|N).P(laugh|N)
  - V N : P(“people laugh”, V N) = P(V).P(N|V).P(people|V).P(laugh|N)
  - V V : P(“people laugh”, V V) = P(V).P(V|V).P(people|V).P(laugh|V)

Forward Probability

- ^ People laugh $
- ^ N N $

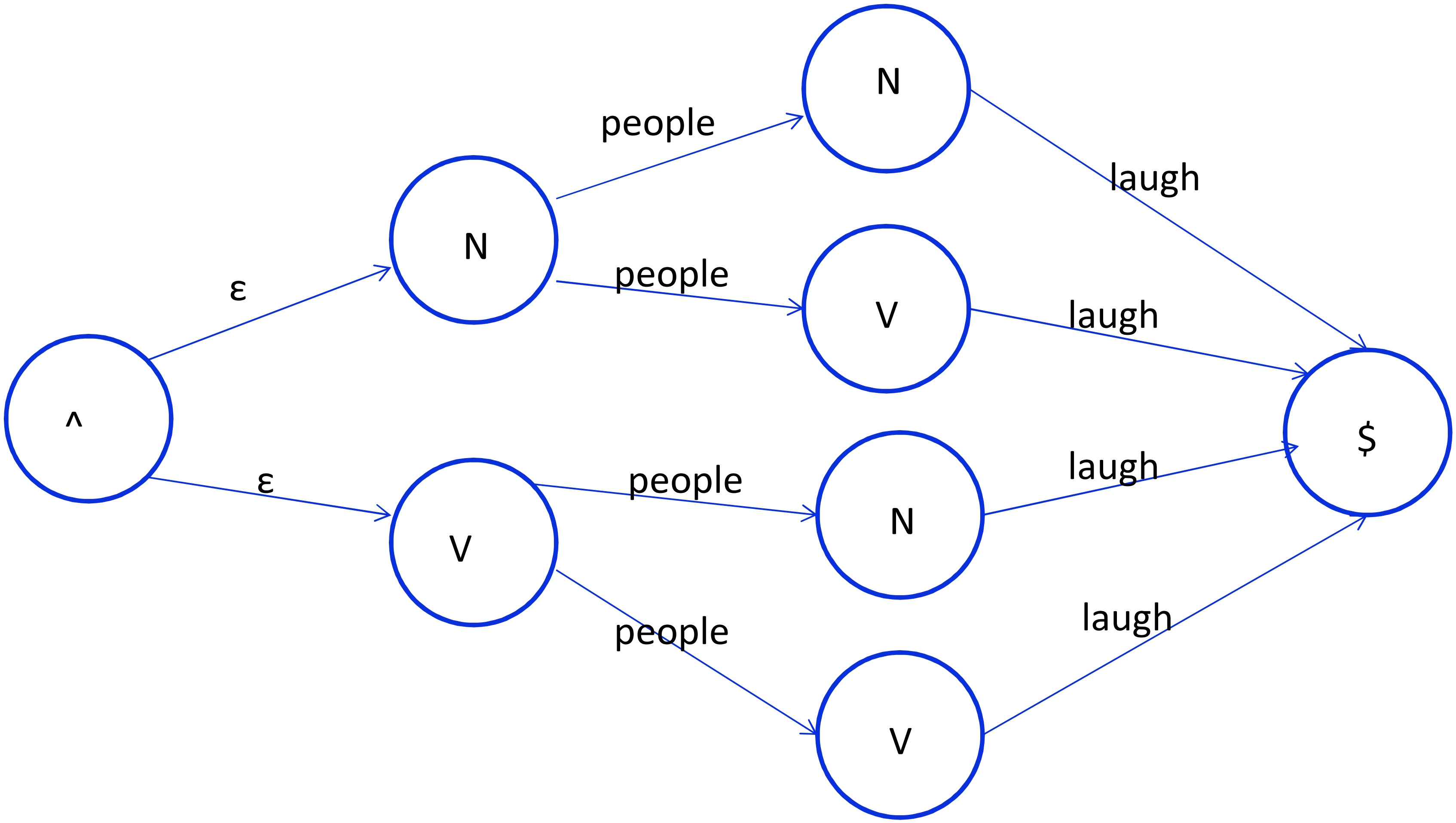
  V V

P(People laugh)

= F(People laugh, $)

= F(People, N).P(N->^{laugh}$)+F(people, V).P(V->^{laugh}$)

Label Sequence to Automaton



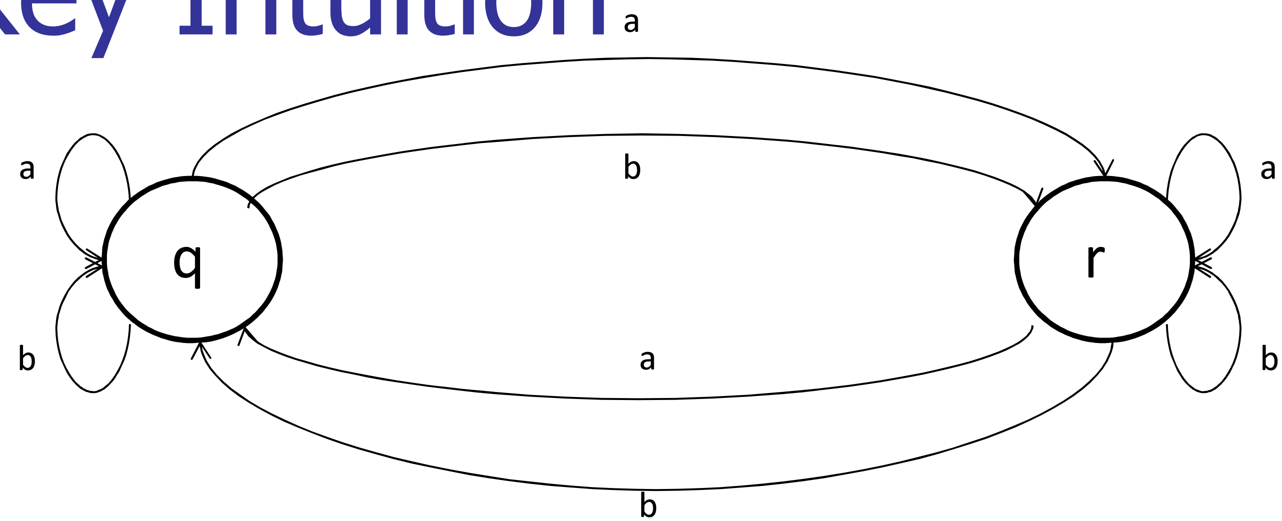
Forward Probability (contd.)

- P(people laugh) = F(people laugh, \$)
  = F(people, N).P(N->^{laugh}\$)+F(people, V).P(V->^{laugh}\$)
- F(people, N)=F(^ people, N)
  =F(^, N).P(N->^{people}N)+F(^, V).P(V->^{people}N)
- F(people, V)=F(^ people, V)
  =F(^, N).P(N->^{people}V)+F(^, V).P(V->^{people}V)

HMM Training

Baum Welch or Forward Backward
Algorithm

Key Intuition



Given: Training sequence

Initialization: Probability values

Compute: Pr (state seq | training seq)

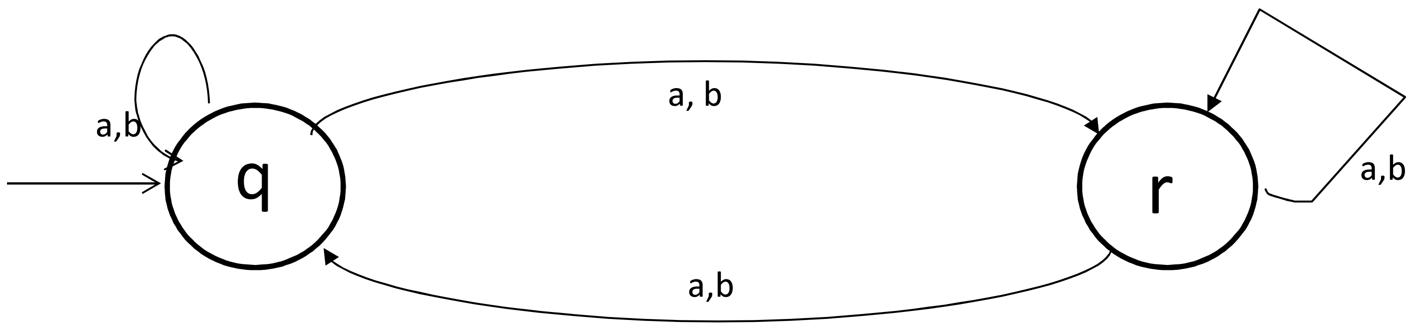
get expected count of transition

compute rule probabilities

Approach: Initialize the probabilities and recompute them…

EM like approach

Baum-Welch algorithm: counts



String = abb aaa bbb aaa

Sequence of states with respect to input symbols

o/p seq

$$\rightarrow q \xrightarrow{a} r \xrightarrow{b} q \xrightarrow{b} q \xrightarrow{a} r \xrightarrow{a} q \xrightarrow{a} r \xrightarrow{b} q \xrightarrow{b} q \xrightarrow{b} q \xrightarrow{a} r \xrightarrow{a} q \xrightarrow{a} r$$

State seq

Calculating probabilities from table

$$P(q \xrightarrow{a} r) = 5/8$$

$$P(q \xrightarrow{b} r) = 3/8$$

$$P(s^i \xrightarrow{w_k} s^j) = \frac{c(s^i \xrightarrow{w_k} s^j)}{\sum_{l=1}^{T} \sum_{m=1}^{A} c(s^i \xrightarrow{w_m} s^l)}$$

T=#states

A=#alphabet symbols

Table of counts

Src	**Dest**	**O/P**	**Count**
q	r	a	5
q	q	b	3
r	q	a	3
r	q	b	2

Now if we have a non-deterministic transitions then multiple state seq possible for the given o/p seq (ref. to previous slide's feature). Our aim is to find expected count through this.

Interplay Between Two Equations

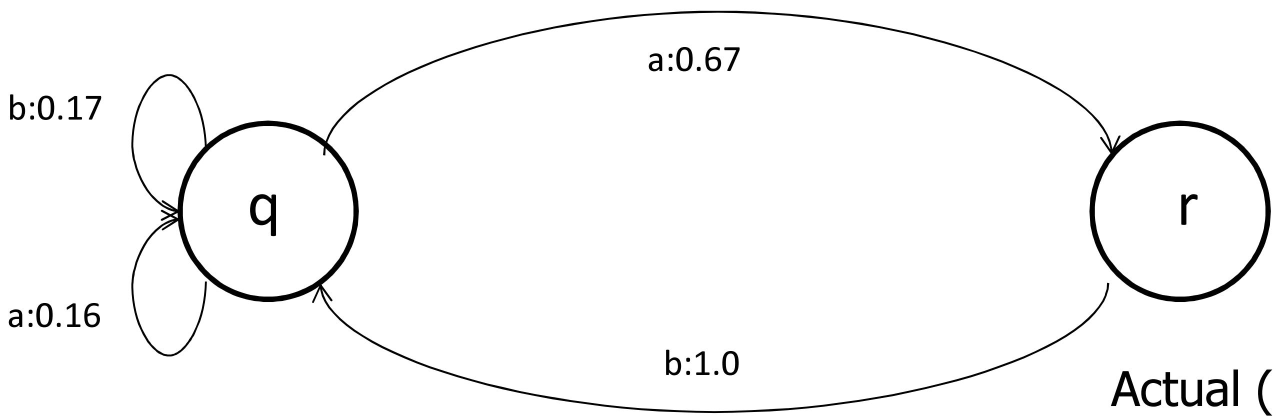
$$P(s^i \xrightarrow{W_k} s^j) = \frac{c(s^i \xrightarrow{W_k} s^j)}{\sum_{l=0}^{T} \sum_{m=0}^{A} c(s^i \xrightarrow{W_m} s^l)}$$

$$C(s^i \xrightarrow{W_k} s^j) =$$

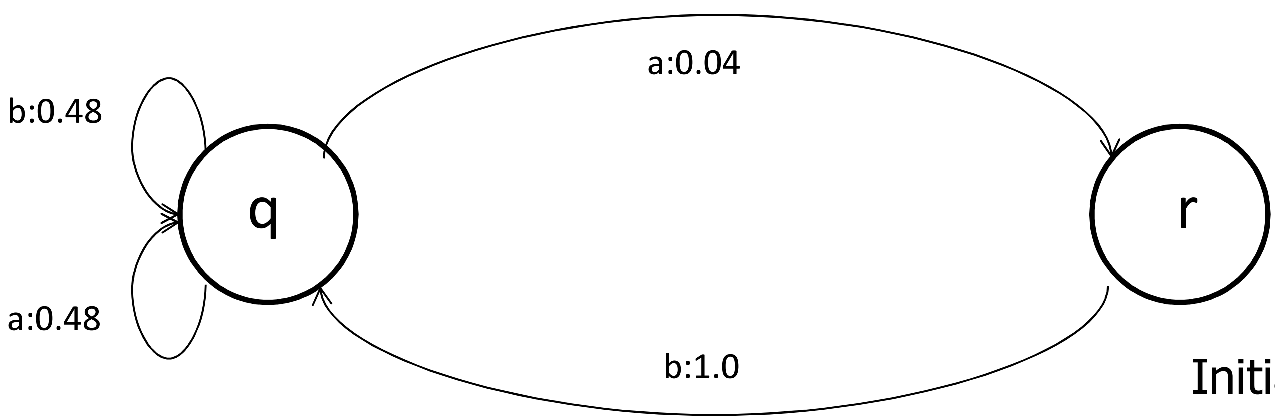
$$\sum_{s_{0,n+1}} P(S_{0,n+1} \mid W_{0,n}) \times n(s^i \xrightarrow{W_k} s^j, S_{0,n+1}, w_{0,n})$$

No. of times the transitions $s^i \xrightarrow{w_k} s^j$ occurs in the string

Illustration



Actual (Desired) HMM



Initial guess

One run of Baum-Welch algorithm: *string ababb*

$\in \to a$	$a \to b$	$b \to a$	$a \to b$	$b \to b$	$b \to \in$	P(path)	$q \xrightarrow{a} r$	$r \xrightarrow{b} q$	$q \xrightarrow{a} q$	$q \xrightarrow{b} q$
q	r	q	r	q	q	0.00077	0.00154	0.00154	0	0.00077
q	r	q	q	q	q	0.00442	0.00442	0.00442	0.00442	0.00884
q	q	q	r	q	q	0.00442	0.00442	0.00442	0.00442	0.00884
q	q	q	q	q	q	0.02548	0.0	0.000	0.05096	0.07644
Rounded Total →						0.035	0.01	0.01	0.06	0.095
New Probabilities (P) → State sequences							0.06 =(0.01/(0.01+0.06+0.095)	1.0	0.36	0.581

* ε is considered as starting and ending symbol of the input sequence string. Through multiple iterations the probability values will converge.

Computational part (1/2)

$$C(s^i \xrightarrow{w_k} s^j) = \sum_{s_{0,n+1}} [P(S_{0,n+1} \mid W_{0,n}) \times n(s^i \xrightarrow{w_k} s^j, S_{0,n+1}, W_{0,n})]$$

$$= \frac{1}{P(W_{0,n})} \sum_{s_{0,n+1}} [P(S_{0,n+1}, W_{0,n}) \times n(s^i \xrightarrow{w_k} s^j, S_{0,n+1}, W_{0,n})]$$

$$= \frac{1}{P(W_{0,n})} \sum_{t=0,n} \sum_{s_{0,n+1}} [P(S_t = s^i, W_t = w_k, S_{t+1} = s^j, S_{0,n+1}, W_{0,n})]$$

$$= \frac{1}{P(W_{0,n})} \sum_{t=0,n} [P(S_t = s^i, W_t = w_k, S_{t+1} = s^j, W_{0,n})]$$

$$S0 \xrightarrow{w_0} S1 \xrightarrow{w_1} S1 \xrightarrow{w_2} \ldots\ Si \xrightarrow{w_k} Sj \ldots \rightarrow Sn\text{-}1 \xrightarrow{w_{n-1}} Sn \xrightarrow{w_n} Sn+1$$

Computational part (2/2)

$$\sum_{t=0}^{n} P(S_t = s^i, S_{t+1} = s^j, W_t = w_k, W_{0,n})$$

$$= \sum_{t=0}^{n} P(W_{0,t-1}, S_t = s^i, S_{t+1} = s^j, W_t = w_k, W_{t+1,n})$$

$$= \sum_{t=0}^{n} P(W_{0,t-1}, S_t = s^i) P(S_{t+1} = s^j, W_t = w_k \mid W_{0,t-1}, S_t = s^i) P(W_{t+1,n} \mid S_{t+1} = s^j)$$

$$= \sum_{t=0}^{n} F(t-1, i) P(S_{t+1} = s^j, W_t = w_k \mid S_t = s^i) B(t+1, j)$$

$$= \sum_{t=0}^{n} F(t-1, i) P(S_{t+1} = s^j, W_t = w_k \mid S_t = s^i) B(t+1, j)$$

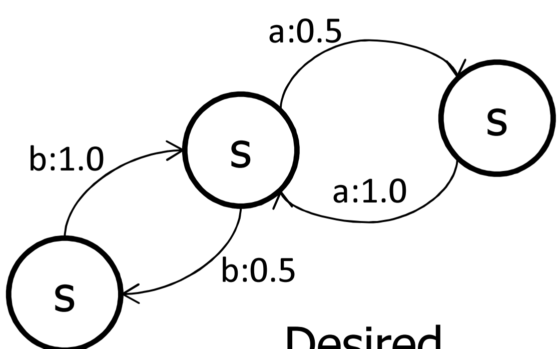
$$= \sum_{t=0}^{n} F(t-1, i) P(s^i \xrightarrow{w_k} s^j) B(t+1, j)$$

$$S0 \xrightarrow{w_0} S1 \xrightarrow{w_1} S1 \xrightarrow{w_2} \ldots\ Si \xrightarrow{w_k} Sj \ldots \rightarrow Sn\text{-}1 \xrightarrow{w_{n-1}} Sn \xrightarrow{w_n} Sn+1$$

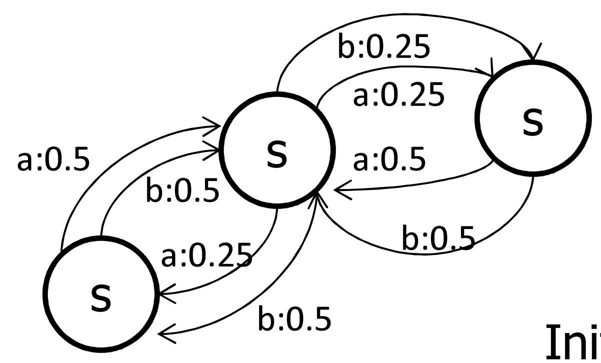
Discussions

1. Symmetry breaking:

   Example: Symmetry breaking leads to no change in initial values



Desired



Initialized

2 Struck in Local maxima

3. Label bias problem

   Probabilities have to sum to 1.

   Values can rise at the cost of fall of values for others.

Indian Language POS tag standard

Sl. No	Category			Label	Annotation Convention**	Remarks
	Top level	Subtype (level 1)	Subtype (level 2)			
1	**Noun**			**N**	**N**	
1.1		Common		NN	N__NN	
1.2		Proper		NNP	N__NNP	
1.3		Verbal		NNV	N__NNV	The verbal noun type is only for languages such as Tamil and Malyalam)
1.4		Nloc		NST	N__NST	
2	**Pronoun**			**PR**	**PR**	
2.1		Personal		PRP	PR__PRP	
2.2		Reflexive		PRF	PR__PRF	
2.3		Relative		PRL	PR__PRL	
2.4		Reciprocal		PRC	PR__PRC	
2.5		Wh-word		PRQ	PR__PRQ	
3	**Demonstrative**			**DM**	**DM**	
3.1		Deictic		DMD	DM__DMD	
3.2		Relative		DMR	DM__DMR	
3.3		Wh-word		DMQ	DM__DMQ	
4	**Verb**			**V**	**V**	
4.1		Main		VM	V__VM	
04/01/01			Finite	VF	V__VM__VF	
04/01/02			Non-finite	VNF	V__VM__VNF	
04/01/03			Infinitive	VINF	V__VM__VINF	
04/01/04			Gerund	VNG	V__VM__VNG	
4.2		Auxiliary		VAUX	V__VAUX	
5	**Adjective**			**JJ**		
6	**Adverb**			**RB**		Only manner adverbs
7	**Postposition**			**PSP**		

8	**Conjunction**			**CC**	**CC**	
8.1		Co-ordinator		CCD	CC__CCD	
8.2		Subordinator		CCS	CC__CCS	
08/02/ 01			Quotative	UT	CC__CCS__UT	
9	**Particles**			**RP**	**RP**	
9.1		Default		RPD	RP__RPD	
9.2		Classifier		CL	RP__CL	
9.3		Interjection		INJ	RP__INJ	
9.4		Intensifier		INTF	RP__INTF	
9.5		Negation		NEG	RP__NEG	
10	**Quantifiers**			**QT**	**QT**	
10.1		General		QTF	QT__QTF	
10.2		Cardinals		QTC	QT__QTC	
10.3		Ordinals		QTO	QT__QTO	
11	**Residuals**			**RD**	**RD**	
11.1		Foreign word		RDF	RD__RDF	A word written in script other than the script of the original text
11.2		Symbol		SYM	RD__SYM	For symbols such as $, & etc
11.3		Punctuation		PUNC	RD__PUNC	Only for punctuations
11.4		Unknown		UNK	RD__UNK	
11.5		Echowords		ECH	RD__ECH	

** The annotation is to be done using the lowest level tag of the type hierarchy. Once the lower level tag is selected, the higher level tags should be stored automatically.