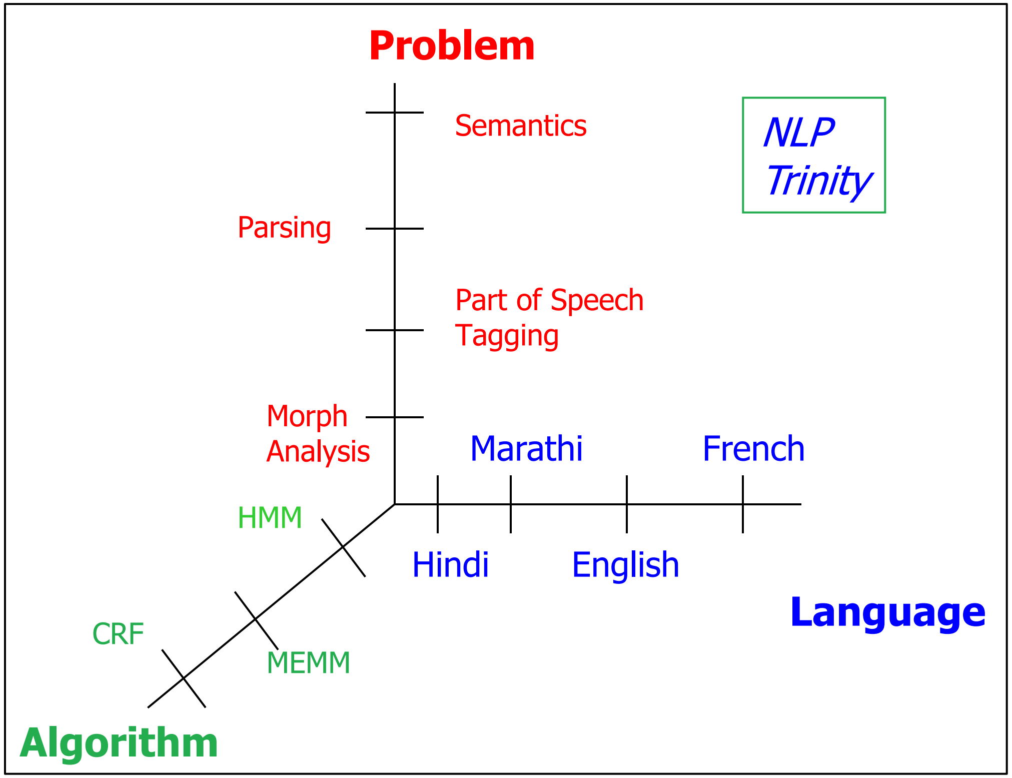
# Part of Speech Tagging

Asif Ekbal
CSE Dept.,
IIT Patna

# Part of Speech Tagging

With Hidden Markov Model

# NLP Trinity


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

# Part of Speech Tagging

- POS Tagging: attaches to each word in a sentence a part of speech tag from a given set of tags called the **Tag-Set**
- Standard Tag-set : Penn Treebank (for English)- *36 tags*

# Example

- “_“ The_DT mechanisms_NNS that_WDT make_VBP traditional_JJ hardware_NN are_VBP really_RB being_VBG obsoleted_VBN by_IN microprocessor-based_JJ machines_NNS ,_, ”_” said_VBD Mr._NNP Benton_NNP ._.

# Where does POS tagging fit in

Increased Complexity Of Processing

Discourse and Corefernce

Semantics Extraction

Parsing

Chunking

POS tagging

Morphology

# Example to illustrate complexity of POS taggng

# POS tagging is disambiguation

*N (noun), V (verb), J (adjective), R (adverb) and F (other, i.e., function words).*

***That_F former_J Sri_Lanka_N skipper_N and_F ace_J batsman_N Aravinda_De_Silva_N is_F a_F man_N of_F few_J words_N was_F very_R much_R evident_J on_F Wednesday_N when_F the_F legendary_J batsman_N ,_F who_F has_V always_R let_V his_N bat_N talk_V ,_F struggled_V to_F answer_V a_F barrage_N of_F questions_N at_F a_F function_N to_F promote_V the_F cricket_N league_N in_F the_F city_N ._F***

# POS disambiguation

- ***That_F/N/J*** *('that' can be complementizer (can be put under 'F'), demonstrative (can be put under 'J') or pronoun (can be put under 'N'))*
- ***former_J***
- ***Sri_N/J Lanka_N/J*** *(Sri Lanka together qualify the skipper)*
- ***skipper_N/V*** *('skipper' can be a verb too)*
- ***and_F***
- ***ace_J/N*** *('ace' can be both J and N; "Nadal served an ace")*
- ***batsman_N/J*** *('batsman' can be J as it qualifies Aravinda De Silva)*
- ***Aravinda_N De_N Silva_N is_F a_F***
- ***man_N/V*** *('man' can verb too as in'man the boat')*
- ***of_F few_J***
- ***words_N/V*** *('words' can be verb too, as in 'he words is speeches beautifully')*

# Behaviour of "*That*"

- That
  - *That man is known by the company he keeps.* (Demonstrative)
  - *Man that is known by the company he keeps, gets a good job*. (Pronoun)
  - *That man is known by the company he keeps, is a proverb*. (Complementation)
- Chaotic systems: Systems where a small perturbation in input causes a large change in output

# POS disambiguation

- ***was_F very_R much_R evident_J on_F Wednesday_N***
- ***when_F/N*** *('when' can be a relative pronoun (put under 'N) as in 'I know the time when he comes')*
- ***the_F legendary_J batsman_N***
- ***who_F/N***
- ***has_V always_R let_V his_N***
- ***bat_N/V***
- ***talk_V/N***
- ***struggle_V /N***
- ***answer_V/N***
- ***barrage_N/V***
- ***question_N/V***
- ***function_N/V***
- ***promote_V cricket_N league_N city_N***

# Mathematics of POS tagging

# Argmax computation (1/2)

Best tag sequence
= $T^*$
= argmax $P(T|W)$
= argmax $P(T)P(W|T)$ (by Baye's Theorem)

$$P(T) = P(t_0 t_1 t_2 \ldots t_{n+1})$$
$$= P(t_0)P(t_1|t_0)P(t_2|t_1 t_0)P(t_3|t_2 t_1 t_0) \ldots P(t_n|t_{n-1} t_{n-2} \ldots t_0)P(t_{n+1}|t_n t_{n-1} \ldots t_0)$$
$$= P(t_0)P(t_1|t_0)P(t_2|t_1) \ldots P(t_n|t_{n-1})P(t_{n+1}|t_n)$$

$$= \prod_{i=0}^{N+1} P(t_i|t_{i-1})$$ Bigram Assumption

# Argmax computation (2/2)

$$P(W|T) = P(w_0|t_0\text{-}t_{n+1})P(w_1|w_0t_0\text{-}t_{n+1})P(w_2|w_1w_0t_0\text{-}t_{n+1}) \ldots$$
$$P(w_n|w_0\text{-}w_{n-1}t_0\text{-}t_{n+1})P(w_{n+1}|w_0\text{-}w_nt_0\text{-}t_{n+1})$$

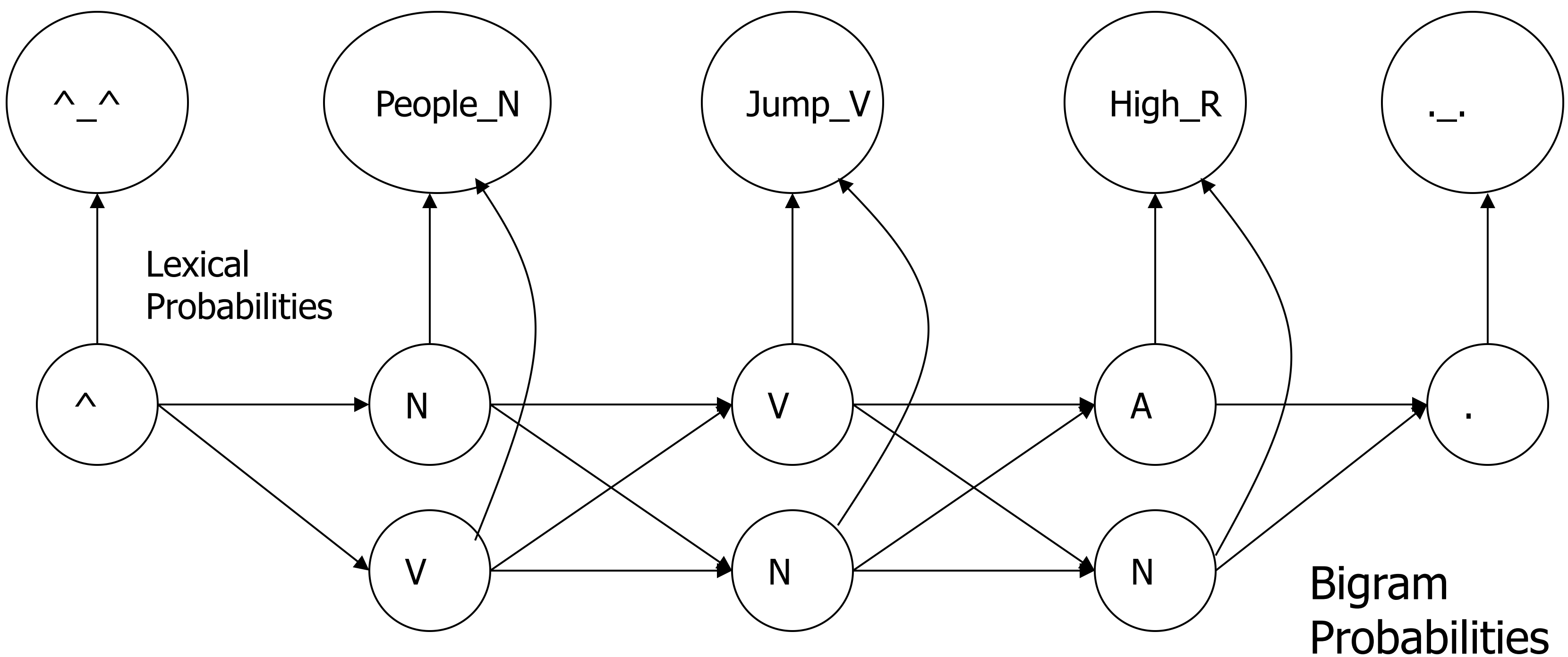
Assumption: A word is determined completely by its tag. This is inspired by speech recognition

$$= P(w_o|t_o)P(w_1|t_1) \ldots P(w_{n+1}|t_{n+1})$$

$$= \prod_{i=0}^{n+1} P(w_i|t_i)$$

$$= \prod_{i=1}^{n+1} P(w_i|t_i) \qquad \text{(Lexical Probability Assumption)}$$

# Generative Model



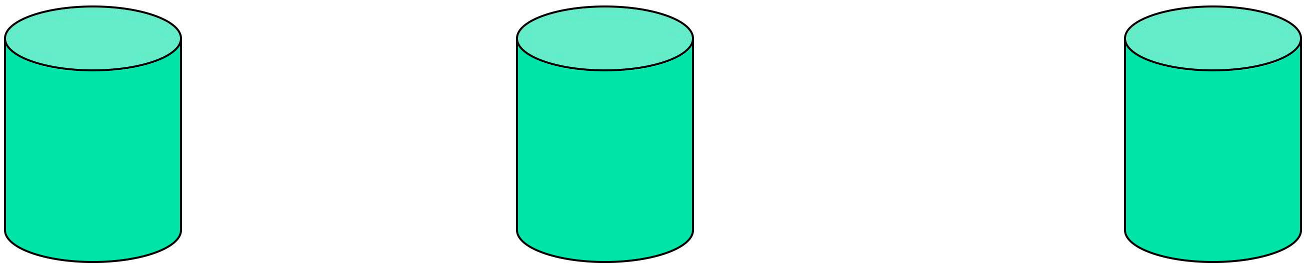

This model is called Generative model.
Here words are observed from tags as states.
This is similar to HMM.

# *Typical POS tag steps*

- Implementation of Viterbi – Unigram, Bigram.
- Evaluation
- Per POS Accuracy
- Confusion Matrix

# A Motivating 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 :

| | $U_1$ | $U_2$ | $U_3$ |
|---|---|---|---|
| $U_1$ | 0.1 | 0.4 | 0.5 |
| $U_2$ | 0.6 | 0.2 | 0.2 |
| $U_3$ | 0.3 | 0.4 | 0.3 |

Transition probability table

and

| | 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 |

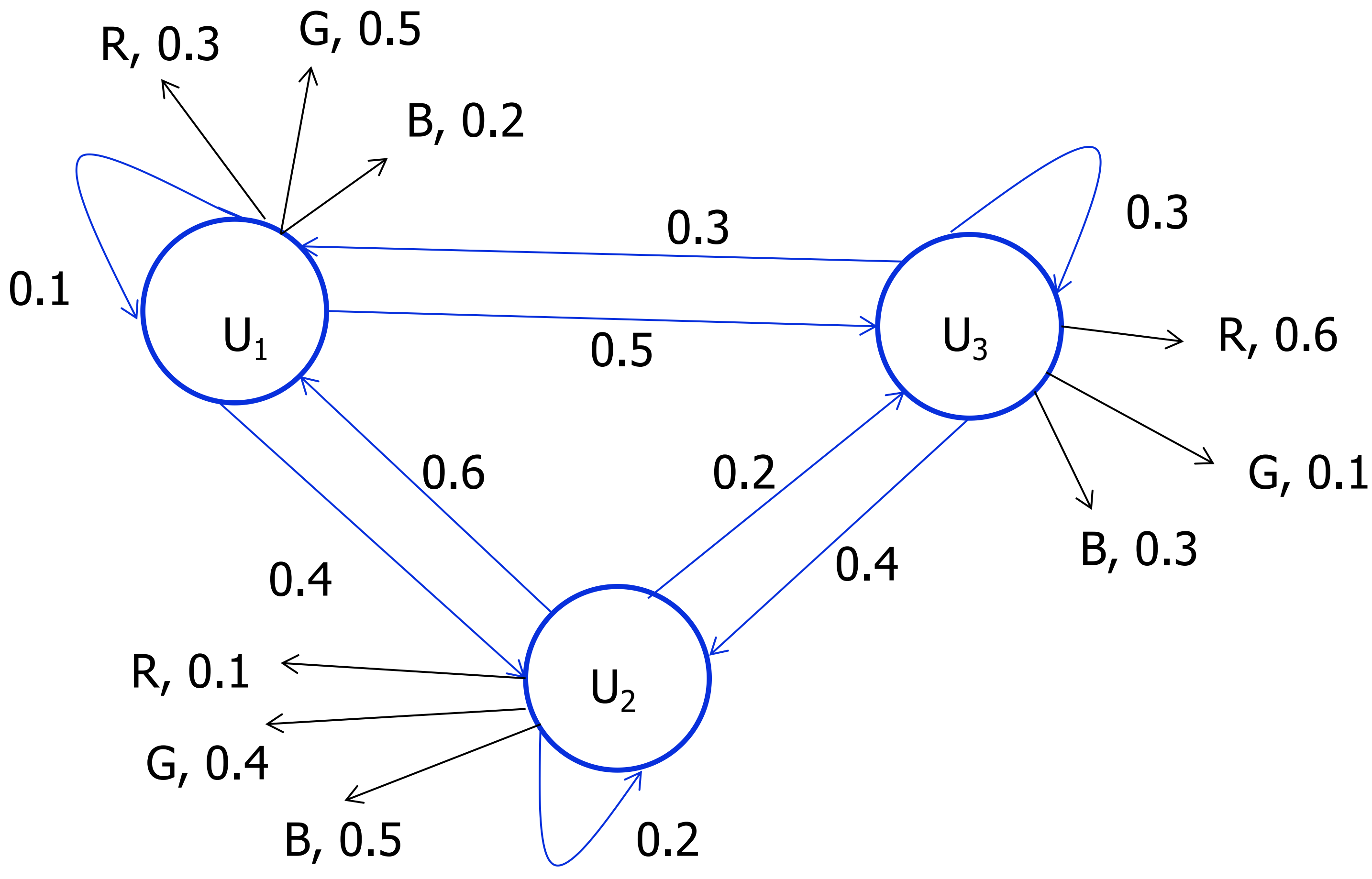
Emission probability table

Observation : RRGGBRGR

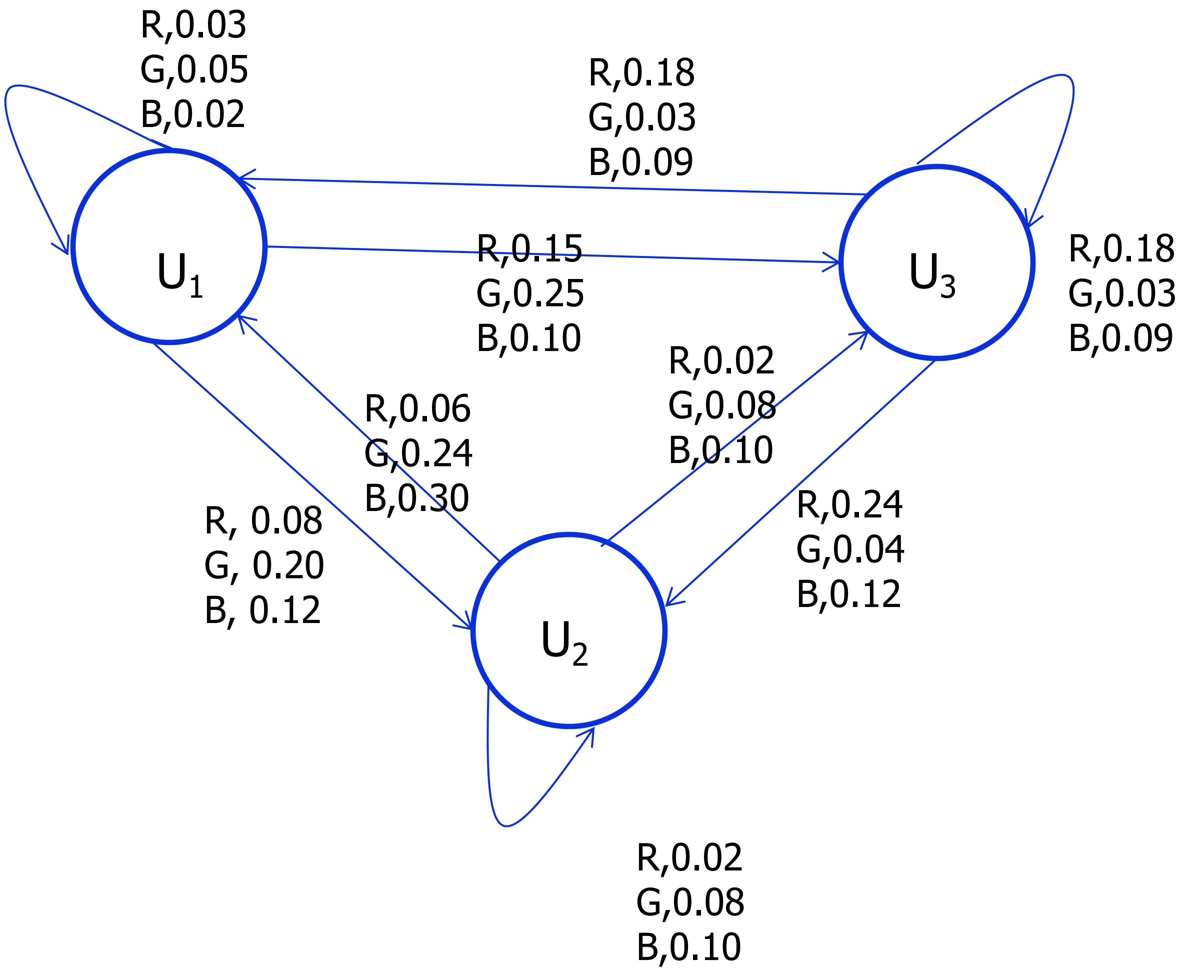
State Sequence : ??

Not so Easily Computable.

# Diagrammatic representation (1/2)



# Diagrammatic representation (2/2)



# Classic problems with respect to HMM

1. Given the observation sequence, find the possible state sequences- *Viterbi*
2. Given the observation sequence, find its probability- forward/backward algorithm
3. Given the observation sequence find the HMM parameters.- Baum-Welch algorithm

# Illustration of Viterbi

- The “start” and “end” are important in a sequence
- Subtrees get eliminated due to the Markov Assumption

<u>POS Tagset</u>

- N(noun), V(verb), O(other) [simplified]
- ^ (start), . (end) [start & end states]

# Illustration of Viterbi

Lexicon

people: N, V

laugh: N, V

.

.

.

Corpora for Training

_______ ^ $w_{11}\_t_{11}$ $w_{12}\_t_{12}$ $w_{13}\_t_{13}$ ………………. $w_{1k\_1}\_t_{1k\_1}$ .

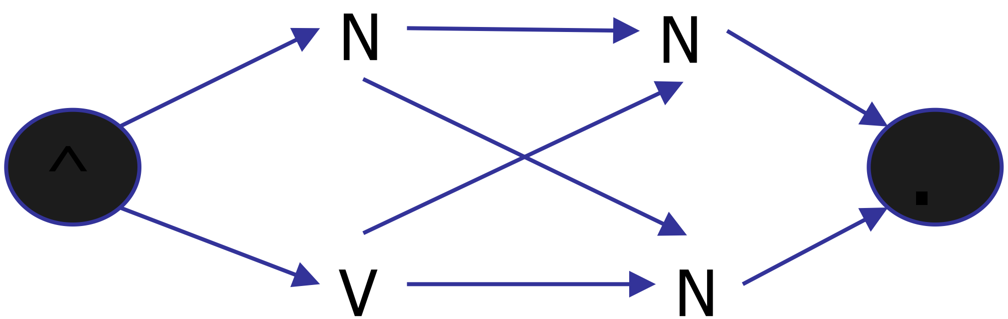
^ $w_{21}\_t_{21}$ $w_{22}\_t_{22}$ $w_{23}\_t_{23}$ ………………. $w_{2k\_2}\_t_{2k\_2}$ .

.

.

^ $w_{n1}\_t_{n1}$ $w_{n2}\_t_{n2}$ $w_{n3}\_t_{n3}$ ………………. $w_{nk\_n}\_t_{nk\_n}$ .

# Inference




Partial sequence graph

| | ^ | N | V | O | . |
|---|---|---|---|---|---|
| ^ | 0 | 0.6 | 0.2 | 0.2 | 0 |
| N | 0 | 0.1 | 0.4 | 0.3 | 0.2 |
| V | 0 | 0.3 | 0.1 | 0.3 | 0.3 |
| O | 0 | 0.3 | 0.2 | 0.3 | 0.2 |
| . | 1 | 0 | 0 | 0 | 0 |

This transition table will change from language to language due to language divergences.

# Lexical Probability Table

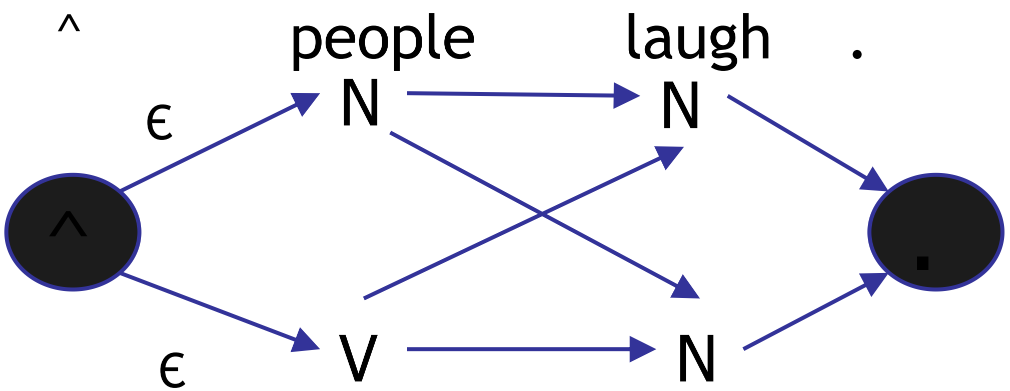
| | Є | people | laugh | … | … |
|---|---|---|---|---|---|
| ^ | 1 | 0 | 0 | … | 0 |
| N | 0 | $1x10^{-3}$ | $1x10^{-5}$ | … | … |
| V | 0 | $1x10^{-6}$ | $1x10^{-3}$ | … | … |
| O | 0 | 0 | 0 | … | … |
| . | 1 | 0 | 0 | 0 | 0 |

Size of this table = # pos tags in tagset X vocabulary size

vocabulary size = # unique words in corpus

# Inference

New Sentence:




p( ^ N N . | ^ people laugh .)

$= (0.6 \times 1.0) \times (0.1 \times 1 \times 10^{-3}) \times (0.2 \times 1 \times 10^{-5})$

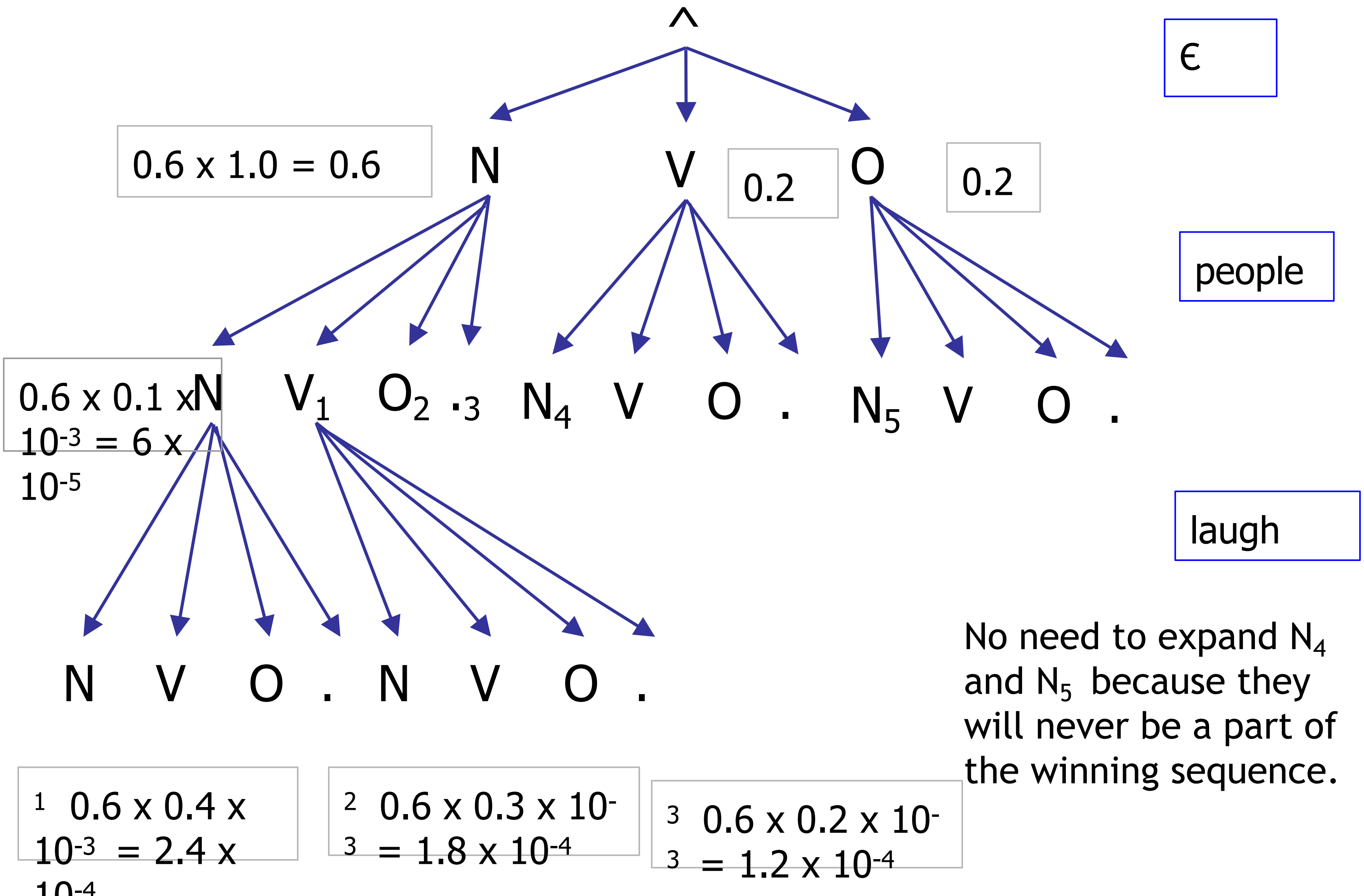
# Computational Complexity

- If we have to get the probability of each sequence and then find maximum among them, we would run into exponential number of computations
- If |s| = #states (tags + ^ + . )

  and |o| = length of sentence ( words + ^ + . )

  Then, #sequences = $s^{|o|-2}$
- But, a large number of partial computations can be reused using Dynamic Programming

# Dynamic Programming




No need to expand $N_4$ and $N_5$ because they will never be a part of the winning sequence.

# Computational Complexity

- Retain only those N / V / O nodes which ends in the highest sequence probability
- Now, complexity reduces from $|s|^{|o|}$ to $|s|.|o|$
- Here, we followed the Markov assumption of order 1

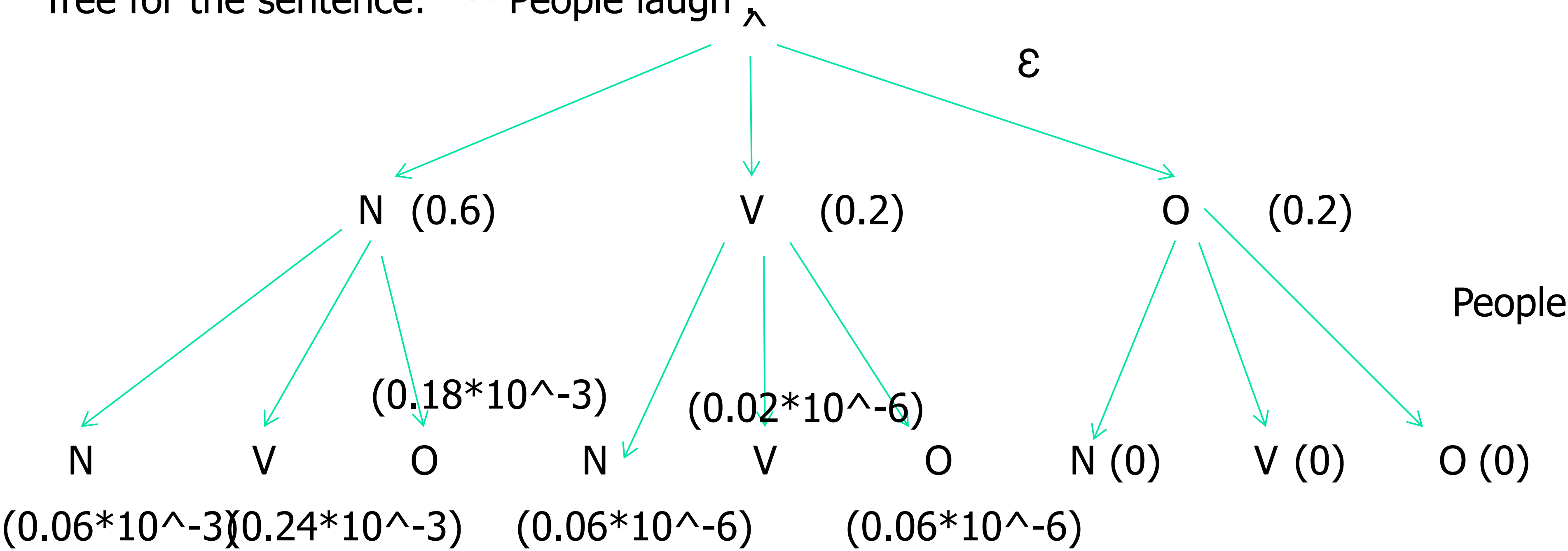
# Points to ponder wrt HMM and Viterbi

# Viterbi Algorithm

- Start with the start state
- Keep advancing sequences that are "maximum" amongst all those ending in the same state

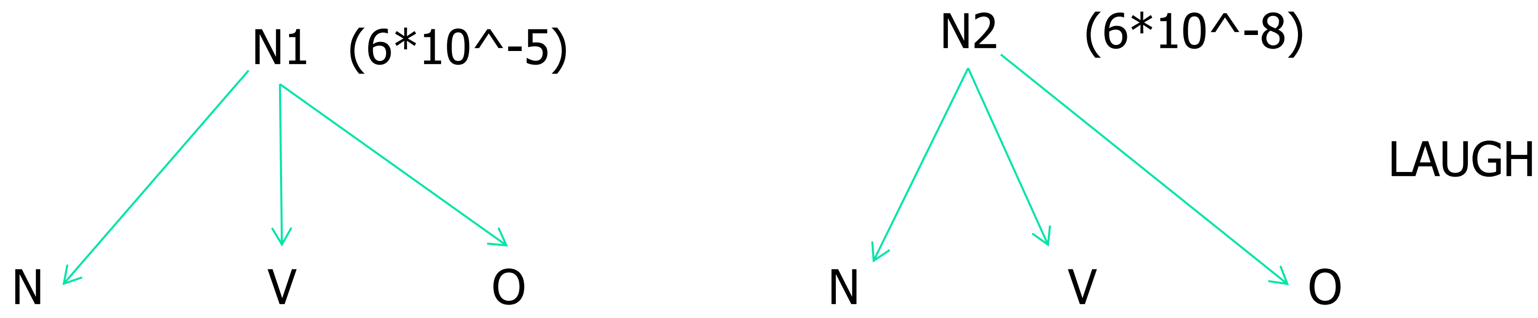
# Viterbi Algorithm

Tree for the sentence: "^ People laugh ."




**Claim**: We do not need to draw all the subtrees in the algorithm

# Viterbi phenomenon (Markov process)




Next step all the probabilities will be multiplied by identical probability (lexical and transition). So children of N2 will have probability less than the children of N1.

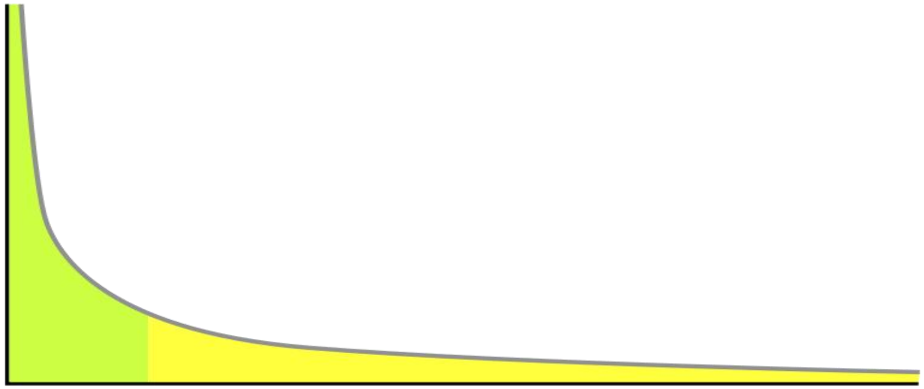
# Effect of shifting probability mass

- Will a word be always given the same tag?
- No. Consider the example:
  - ^ people the city with soldiers .
  - ^ quickly people the city . (i.e., ‘populate’)

- In the first sentence “people” is most likely to be tagged as noun, whereas in the second, probability mass will shift and “people” will be tagged as verb, since it occurs after an adverb.

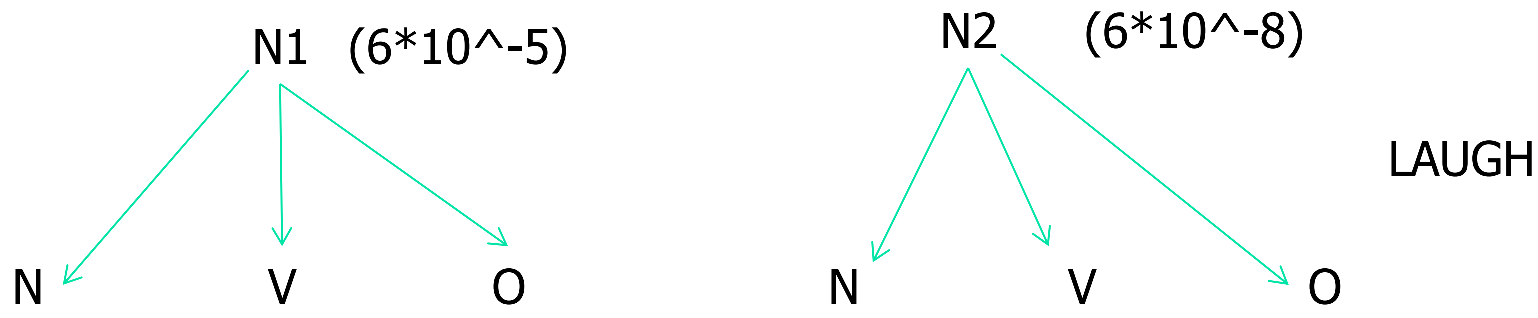
# Tail phenomenon and Language phenomenon

- Long tail Phenomenon: Probability is very low but not zero over a large observed sequence.



- Language Phenomenon:
  - “people” which is predominantly tagged as “Noun” displays a long tail phenomenon.
  - “laugh” is predominantly tagged as “Verb”.

# Viterbi phenomenon (Markov process)




Next step all the probabilities will be multiplied by identical probability (lexical and transition). So children of N2 will have probability less than the children of N1.

# What does $P(A|B)$ mean?

- $P(A|B) = P(B|A)$

  If $P(A)=P(B)$
- $P(A|B)$ means??
  - Causality?? B causes A??
  - Sequentiality?? A follows B?

# Back to the Urn Example

- Here :
  - S = {U1, U2, U3}
  - V = { R,G,B}
- For observation:
  - O ={$o_1$… $o_n$}
- And State sequence
  - Q ={$q_1$… $q_n$}
- π is

$$\pi_i = P(q_1 = U_i)$$

A =

| | $U_1$ | $U_2$ | $U_3$ |
|---|---|---|---|
| $U_1$ | 0.1 | 0.4 | 0.5 |
| $U_2$ | 0.6 | 0.2 | 0.2 |
| $U_3$ | 0.3 | 0.4 | 0.3 |

B=

| | 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 |

# Observations and states

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | $O_6$ | $O_7$ | $O_8$ |
| OBS: | R | R | G | G | B | R | G | R |
| State: | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ | $S_8$ |

$S_i = U_1/U_2/U_3$; A particular state

S: State sequence

O: Observation sequence

S* = "best" possible state (urn) sequence

Goal: Maximize $P(S^*|O)$ by choosing "best" S

# Goal

- Maximize P(S|O) where S is the State Sequence and O is the Observation Sequence

$$S^* = \arg\ \max\ _S(P(S \mid O))$$

# False Start

| | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | $O_6$ | $O_7$ | $O_8$ |
|---|---|---|---|---|---|---|---|---|
| OBS: | R | R | G | G | B | R | G | R |
| State: | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ | $S_8$ |

$$P(S \mid O) = P(S_{1-8} \mid O_{1-8})$$

$$P(S \mid O) = P(S_1 \mid O).P(S_2 \mid S_1, O).P(S_3 \mid S_{1-2}, O)\ldots P(S_8 \mid S_{1-7}, O)$$

By Markov Assumption (a state depends only on the previous state)

$$P(S \mid O) = P(S_1 \mid O).P(S_2 \mid S_1, O).P(S_3 \mid S_2, O)\ldots P(S_8 \mid S_7, O)$$

# Baye's Theorem

$$P(A \mid B) = P(A).P(B \mid A) / P(B)$$

P(A) -: Prior
P(B|A) -: Likelihood

$$\operatorname{argmax}_S P(S \mid O) = \operatorname{argmax}_S P(S).P(O \mid S)$$

# State Transitions Probability

$$P(S) = P(S_{1-8})$$

$$P(S) = P(S_1) . P(S_2 | S_1) . P(S_3 | S_{1-2}) . P(S_4 | S_{1-3}) .. P(S_8 | S_{1-7})$$

By Markov Assumption (k=1)

$$P(S) = P(S_1) . P(S_2 | S_1) . P(S_3 | S_2) . P(S_4 | S_3) .. P(S_8 | S_7)$$

# Observation Sequence probability

$$P(O|S) = P(O_1|S_{1-8}).P(O_2|O_1,S_{1-8}).P(O_3|O_{1-2},S_{1-8})..P(O_8|O_{1-7},S_{1-8})$$

Assumption that ball drawn depends only on the Urn chosen

$$P(O|S) = P(O_1|S_1).P(O_2|S_2).P(O_3|S_3)...P(O_8|S_8)$$

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

$$P(S|O) = P(S_1).P(S_2|S_1).P(S_3|S_2).P(S_4|S_3)...P(S_8|S_7).$$

$$P(O_1|S_1).P(O_2|S_2).P(O_3|S_3)...P(O_8|S_8)$$

# Grouping terms

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

$$
\begin{aligned}
& P(S).P(O|S) \\
= \; & [P(O_0|S_0).P(S_1|S_0)]. \\
& [P(O_1|S_1). \quad P(S_2|S_1)]. \\
& [P(O_2|S_2). \quad P(S_3|S_2)]. \\
& [P(O_3|S_3).P(S_4|S_3)]. \\
& [P(O_4|S_4).P(S_5|S_4)]. \\
& [P(O_5|S_5).P(S_6|S_5)]. \\
& [P(O_6|S_6).P(S_7|S_6)]. \\
& [P(O_7|S_7).P(S_8|S_7)]. \\
& [P(O_8|S_8).P(S_9|S_8)].
\end{aligned}
$$

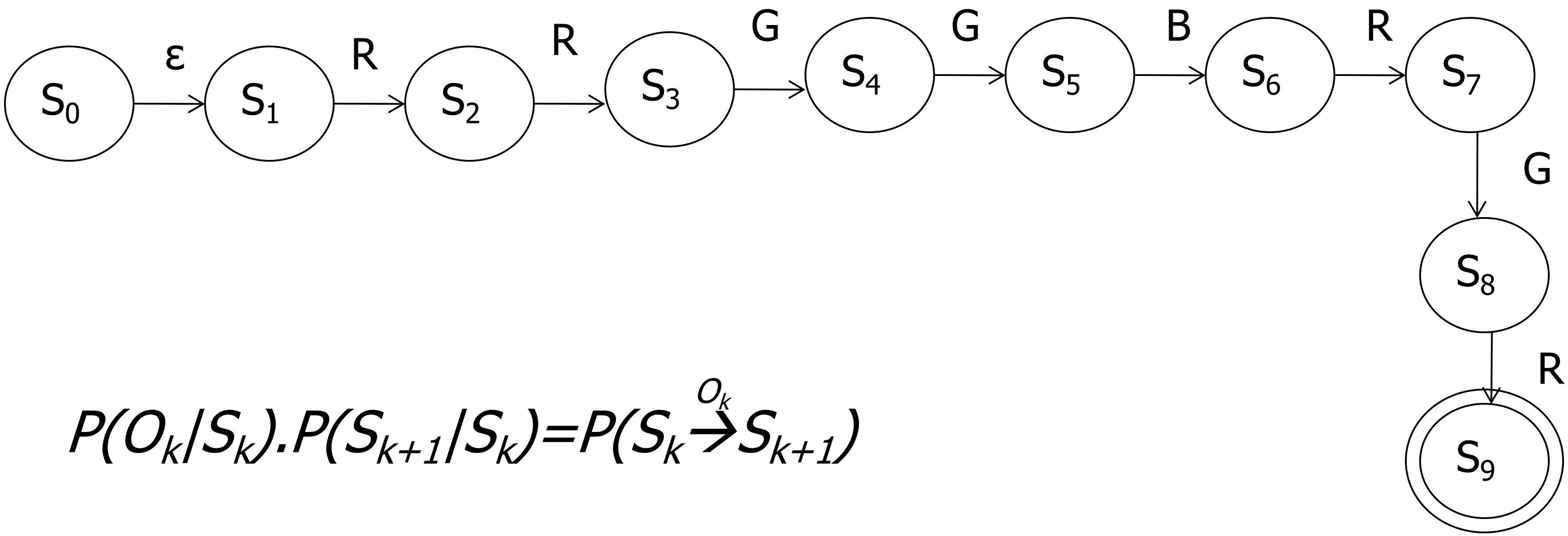
We introduce the states $S_0$ and $S_9$ as initial and final states respectively.

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

$O_0$ is $\varepsilon$-transition

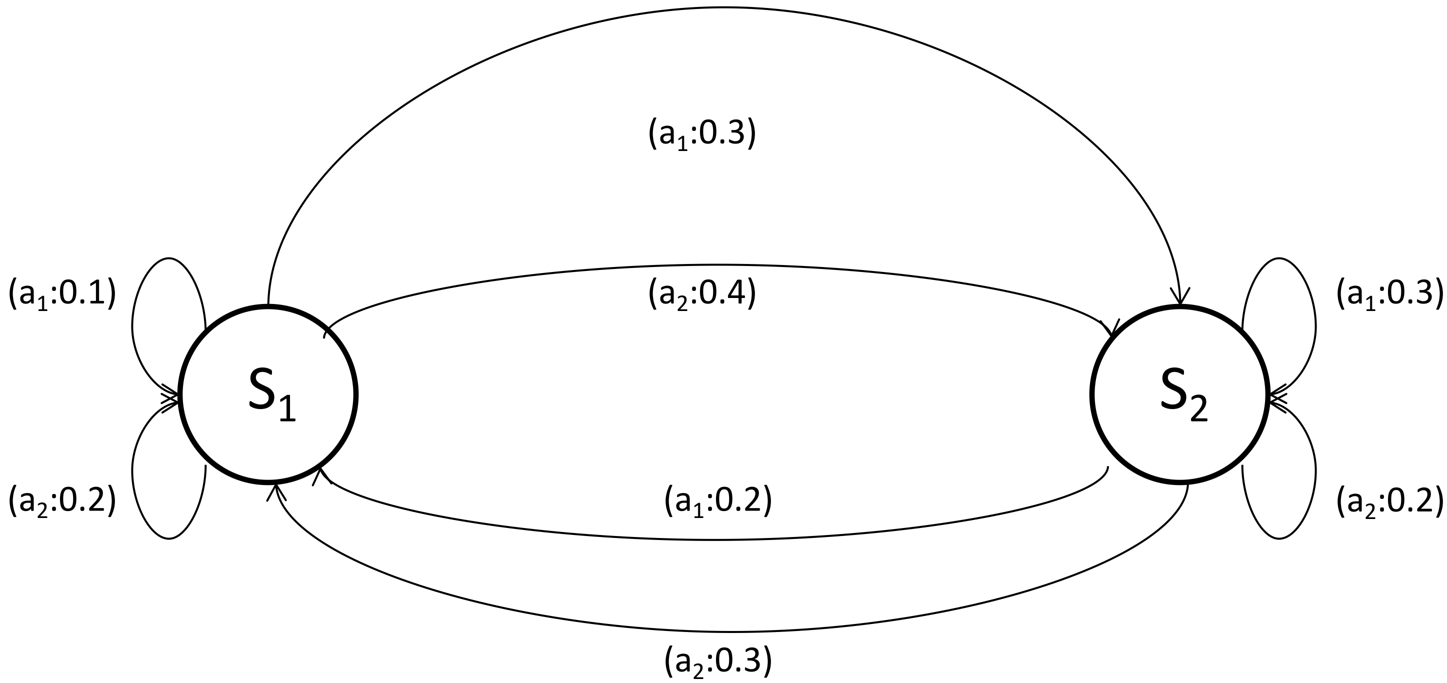
# Introducing useful notation

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



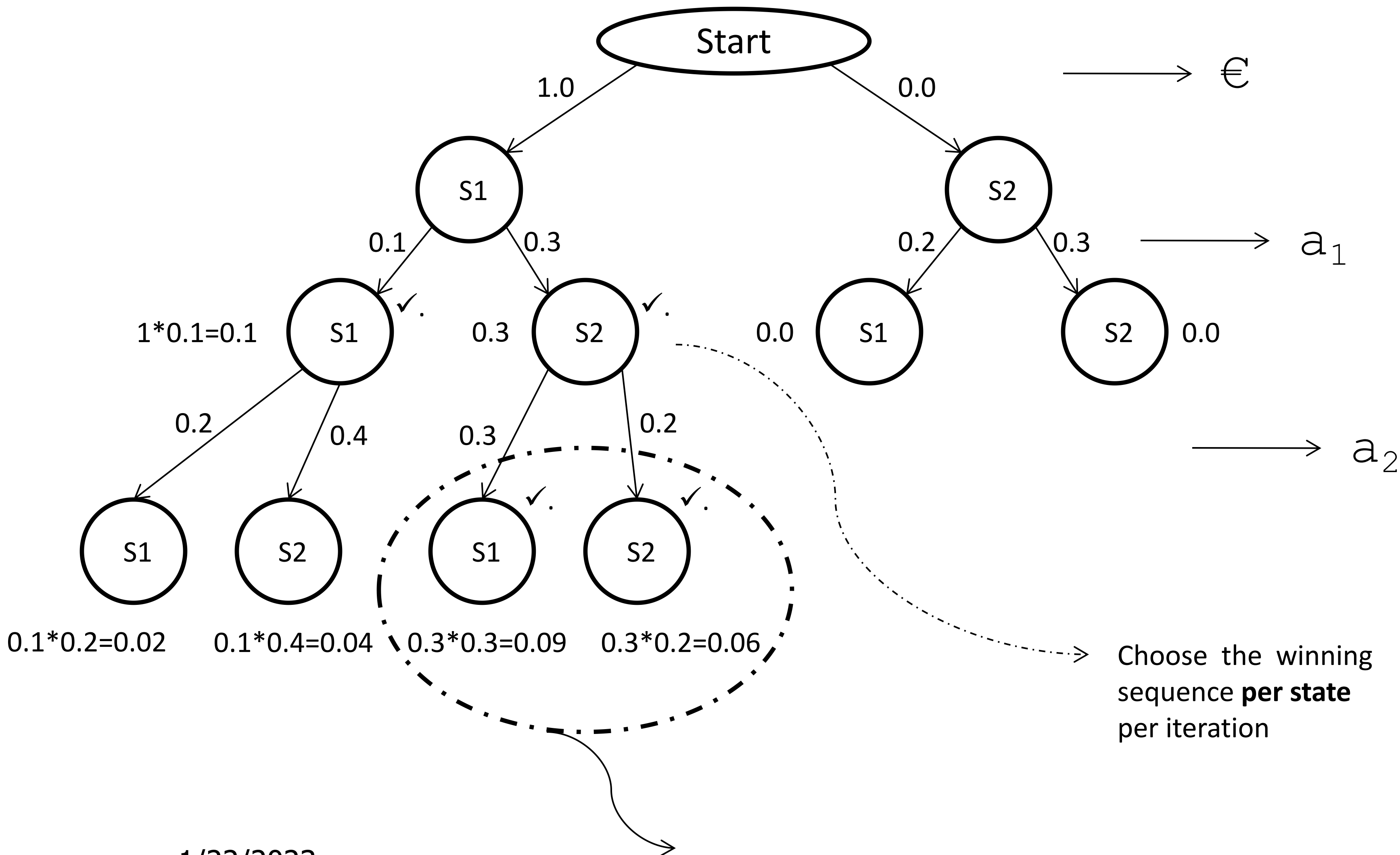

$$P(O_k/S_k).P(S_{k+1}/S_k)=P(S_k \xrightarrow{O_k} S_{k+1})$$

# Probabilistic FSM

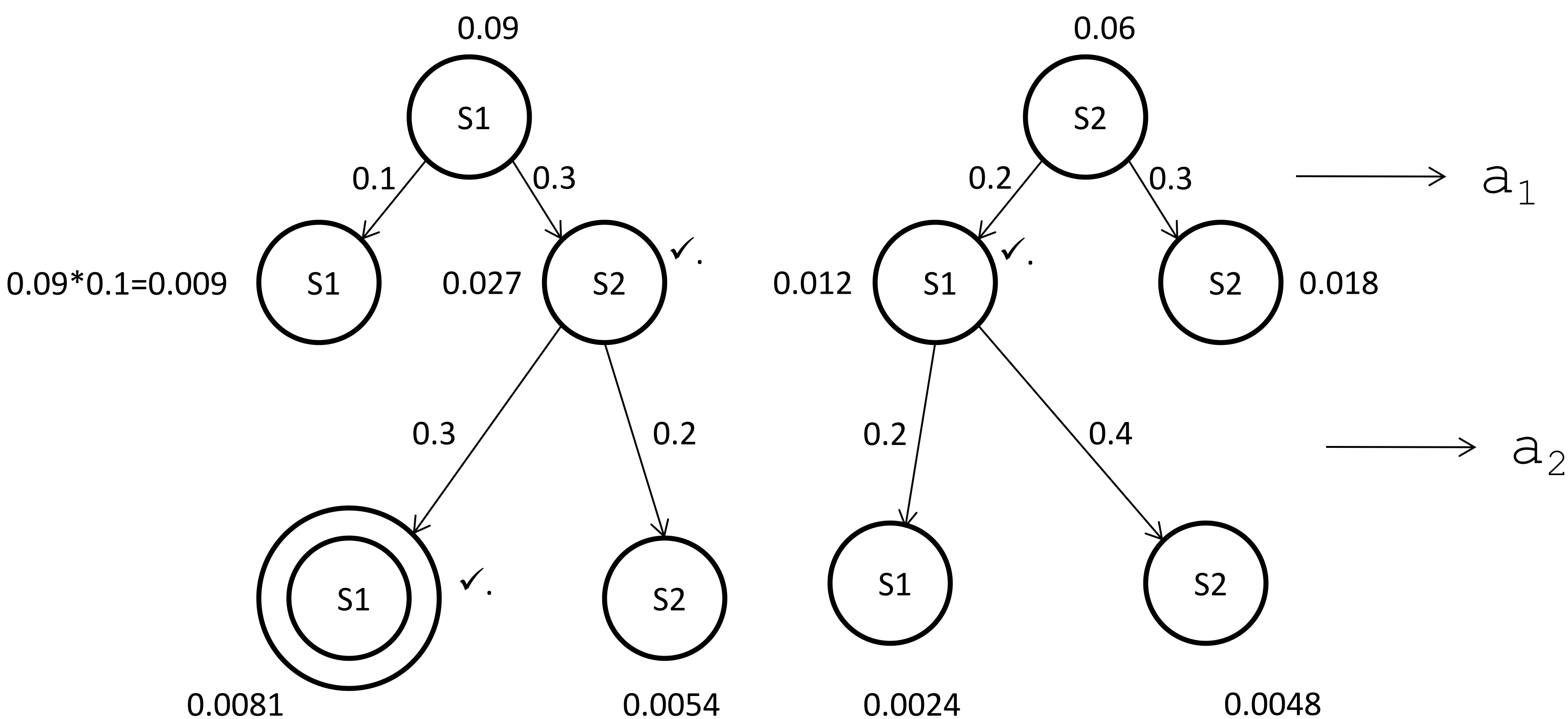



The question here is:
"what is the most likely state sequence given the output sequence seen"

# 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

Path found:
(working backward)

$S_1 \quad S_2 \quad S_1 \quad S_2 \quad S_1$

$a_1 \quad a_2 \quad a_1 \quad a_2$

Problem statement: Find the best possible sequence

$$S^* = \arg\max_{S} P(S \mid O, \mu)$$

$$where,\ S \rightarrow \text{State Seq}, O \rightarrow \text{Output Seq}, \mu \rightarrow \text{Model or Machine}$$

$$\text{Model or Machine} = \{S_0, S, A, T\}$$

Start symbol State collection Alphabet set Transitions

T is defined as $P(S_i \xrightarrow{a_k} S_j) \quad \forall i, j, k$

# Probability of observation sequence

# Why probability of observation sequence?: Language modeling problem

Probabilities computed in the context of corpora

1. P("The sun rises in the east")
2. P("The sun rise in the east")
   - Less probable because of grammatical mistake.
3. P(The svn rises in the east)
   - Less probable because of lexical mistake.
4. P(The sun rises in the west)
   - Less probable because of semantic mistake.

# Uses of language model

1. Detect well-formedness
   - Lexical, syntactic, semantic, pragmatic, discourse
2. Language identification
   - Given a piece of text what language does it belong to.

     *Good morning   - English*

     *Guten morgen   - German*

     *Bon jour          -  French*
3. Automatic speech recognition
4. Machine translation

# How to compute $P(o_0 o_1 o_2 o_3 \ldots o_m)$?

$$P(O) = \sum_{S} P(O, S) \qquad \text{Marginalization}$$

Consider the observation sequence,

$$O_0 O_1 O_2 \ldots\ldots Om$$

$$S_0 \; S_1 \; S_2 \; S_3 .. S_m \; S_{m+1}$$

Where $S_i$ s represent the state sequences.

# Computing $P(o_0o_1o_2o_3\ldots o_m)$

$$
\begin{aligned}
P(O,S) &= P(S)P(O \mid S) \\
&= P(S_0S_1S_2\ldots S_{m+1})P(O_0O_1O_2\ldots O_m \mid S) \\
&= P(S_0).P(S_1 \mid S_0).P(S_2 \mid S_1)\ldots.P(S_{m+1} \mid S_m). \\
&\qquad P(O_0 \mid S_0).P(O_1 \mid S_1)\ldots..P(O_m \mid S_m) \\
&= P(S_0)[P(O_0 \mid S_0).P(S_1 \mid S_0)]\ldots..[P(O_m \mid S_m).P(S_{m+1} \mid S_m)]
\end{aligned}
$$

# Forward and Backward Probability Calculation

# Forward probability $F(k,i)$

- Define $F(k,i)$= Probability of being in state $S_i$ having seen $o_0o_1o_2...o_k$
- $F(k,i)=P(o_0o_1o_2...o_k\,,\ S_i\ )$
- With $m$ as the length of the observed sequence and $N$ states

$$P(observed\ sequence)=P(o_0o_1o_2..o_m)$$
$$=\Sigma_{p=0,N}\ P(o_0o_1o_2..o_m\ ,\ S_p)$$
$$=\Sigma_{p=0,N}\ F(m\ ,\ p)$$

# Forward probability (contd.)

$F(k, q)$
$= P(o_0 o_1 o_2 .. o_k , S_q)$
$= P(o_0 o_1 o_2 .. o_k , S_q)$
$= P(o_0 o_1 o_2 .. o_{k-1} , o_k , S_q)$
$= \Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_{k-1} , S_p , o_k , S_q)$
$= \Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_{k-1} , S_p ).$
$\quad P(o_k , S_q | o_0 o_1 o_2 .. o_{k-1} , S_p)$
$= \Sigma_{p=0,N} F(k-1,p). P(o_k , S_q | S_p)$

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

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

# 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 whole observed sequence
- $P(\text{observed sequence})=P(o_0 o_1 o_2 .. o_m)$

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

  $=B(0,0)$

# Backward probability (contd.)

$B(k\ , p)$

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

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

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

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

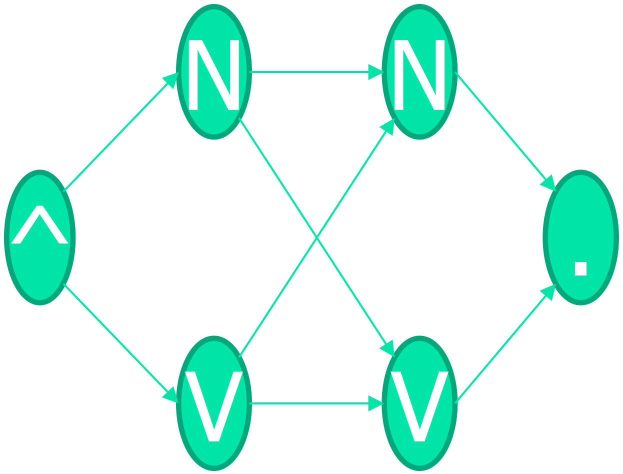
$= \Sigma_{q=0,N}\ P(o_{k+1} o_{k+2} \ldots o_m/S_q\ ).\ P(o_k\ ,$
$\quad S_q/S_p)$

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

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

# How Forward Probability Works

- Goal of Forward Probability: To find P(O) [the probability of Observation Sequence].
- E.g. ^ People laugh .



# Transition and Lexical Probability Tables

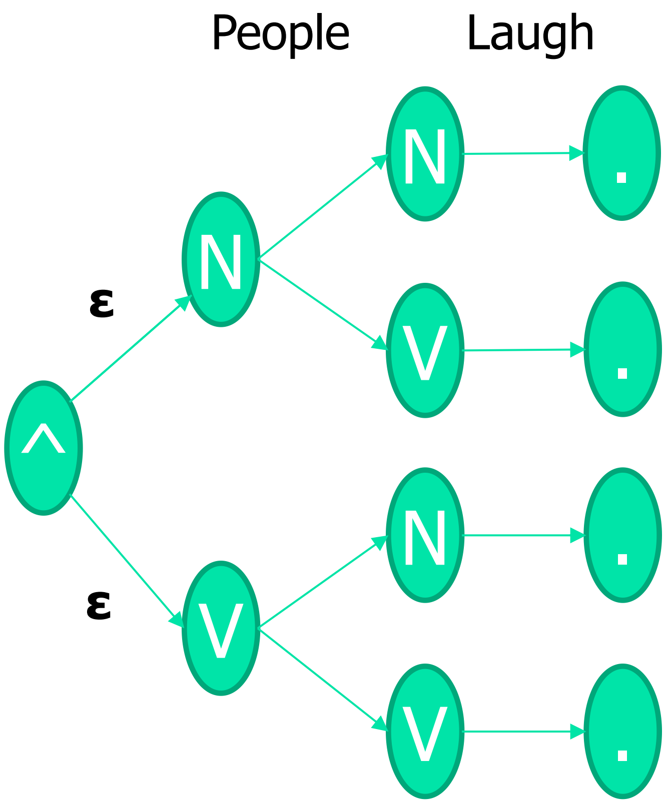
| | ^ | N | V | . |
|---|---|---|---|---|
| ^ | 0 | 0.7 | 0.3 | 0 |
| N | 0 | 0.2 | 0.6 | 0.2 |
| V | 0 | 0.6 | 0.2 | 0.2 |
| . | 1 | 0 | 0 | 0 |

| | ε | People | Laugh |
|---|---|---|---|
| ^ | 1 | 0 | 0 |
| N | 0 | 0.8 | 0.2 |
| V | 0 | 0.1 | 0.9 |
| . | 1 | 0 | 0 |

Inefficient Computation:

$$P(O) = \sum_{S} P(O,S) = \sum_{S} \prod_{i} P(S_i \xrightarrow{o_j} S_j)$$

# Computation in various paths of the Tree




**ε People Laugh**

Path 1: ^ N N .

P(Path1) = (1.0x0.7)x(0.8x0.2)x(0.2x0.2)

**ε People Laugh**

Path 2: ^ N
V .

P(Path2) = (1.0x0.7)x(0.8x0.6)x(0.9x0.2)

**ε People Laugh**

Path 3: ^ V
N .

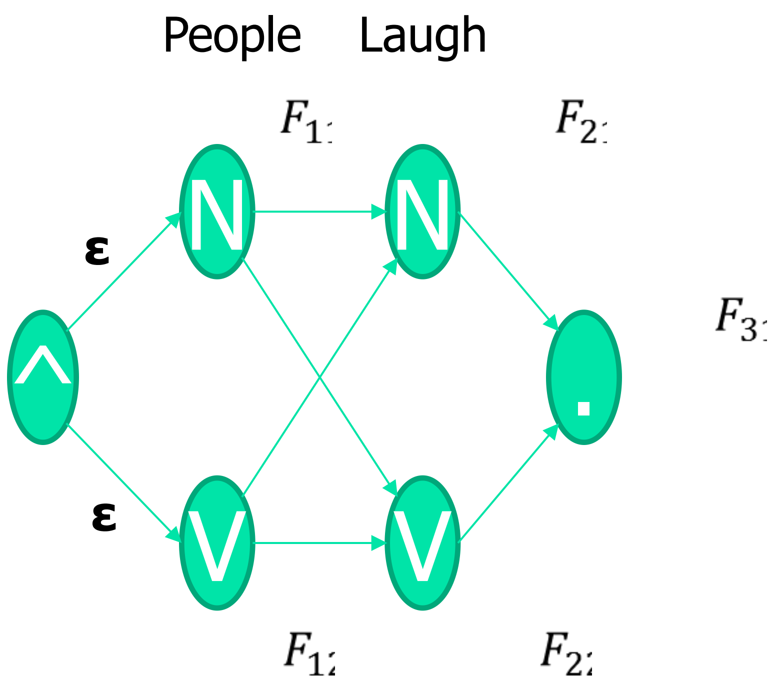
P(Path3) = (1.0x0.3)x(0.1x0.6)x(0.2x0.2)

**ε People Laugh**

Path 4: ^ V
V .

P(Path4) = (1.0x0.3)x(0.1x0.2)x(0.9x0.2)

# Computations on the Trellis




- *F = accumulated F x output probability x transition probability*

$F_{11}$ = 0.7x1.0

$F_{12}$ = 0.3x1.0

$F_{21}$ = $F_{11}$ x (0.2x0.3) + $F_{12}$ x (0.6x0.1)

$F_{22}$ = $F_{11}$ x (0.6x0.8) + $F_{12}$ x (0.2x0.1)

$F_{31}$ = $F_{21}$ x (0.2x0.2) + $F_{22}$ x (0.2x0.9)

# Number of Multiplications

**Tree**

- Each path has 5 multiplications + 1 addition.
- There are 4 paths in the tree.
- Therefore, total of 20 multiplications and 3 additions.

**Trellis**

- $F_{11}$, -> 1 multiplication
- $F_{12}$, -> 1 multiplication
- $F_{21} = F_{11}$ x (1 mult) + $F_{12}$ x (1 mult)

  = 4 multiplications + 1 addition
- Similarly, for $F_{22}$ and $F_{31}$, 4 multiplications and 1 addition each.
- So, total of 14 multiplications and 3 additions.

# Complexity

- Let |S| = #States

And |O| = Observation length - |{^, .}|

- Stage 1 of Trellis: |S| multiplications
- Stage 2 of Trellis: |S| nodes; each node needs computation over |S| arcs.
  - Each Arc = 1 multiplication
  - Accumulated F = 1 more multiplication
  - Total $2|S|^2$ multiplications
- Same for each stage before reading ‘ . ’
- At final stage (‘ . ‘) -> 2|S| multiplications

# Summary : Forward Algorithm

1. Accumulate F over each stage of trellis.
2. Take sum of F values multiplied by $P(S_i \xrightarrow{o_j} S_{i+1})$.
3. Complexity = $|S| + 2|S|^2 (|O| - 1) + 2|S|$

$$= 2|S|^2 |O| - 2|S|^2 + 3|S|$$

$$= O(|S|^2 . |O|)$$

*i.e.*, linear in the length of input and quadratic in number of states.

# Reading List

- TnT (http://www.aclweb.org/anthology-new/A/A00/A00-1031.pdf)
- Brill Tagger (http://delivery.acm.org/10.1145/1080000/1075553/p112-brill.pdf?ip=182.19.16.71&acc=OPEN&CFID=129797466&CFTOKEN=72601926&__acm__=1342975719_082233e0ca9b5d1d67a9997c03a649d1)