CS626: Speech, NLP and Web

POS Tagging cntd, Viterbi, evaluation

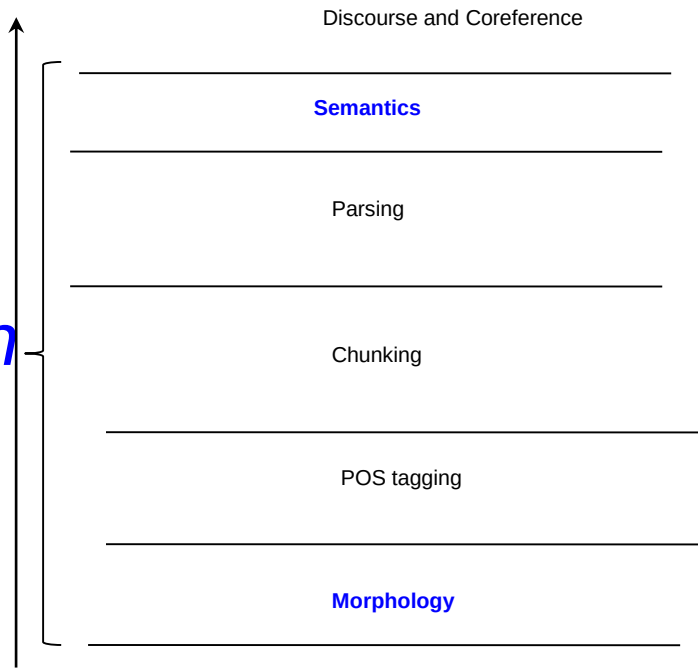
Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

Week 3 of 12th August, 2024

(only one lecture done due to 15th Thursday being a holiday)

1-slide recap of week of 5th Aug

- Covered NLP stack with chatGPT's performance at each layer; chatGPT is an LLM based CAI
  - *To bank, I bank on the bank on the river bank*
- POS tag definition and argmax based formulation
- Should we apply Bayes theorem or not- discriminative (LHS of Argmax) vs. generative (RHS)
- HMM as the apt technique for POS



Should I apply Bayes Rule?

Cancer Detection vs. Visa Granting

- *P(B|A)=[P(B).P(A|B)]/P(A)*
- *Cancer, if P(Cancer) > P(~cancer), i.e. P(cancer)>0.5*

- *Grant_Visa, if P(Grant_Visa) > P(~Grant_Visa), i.e. P(Grant_Visa)>0.5*

- Key consideration- which data is more reliable for obtaining probabilities

Probabilities involved: cancer

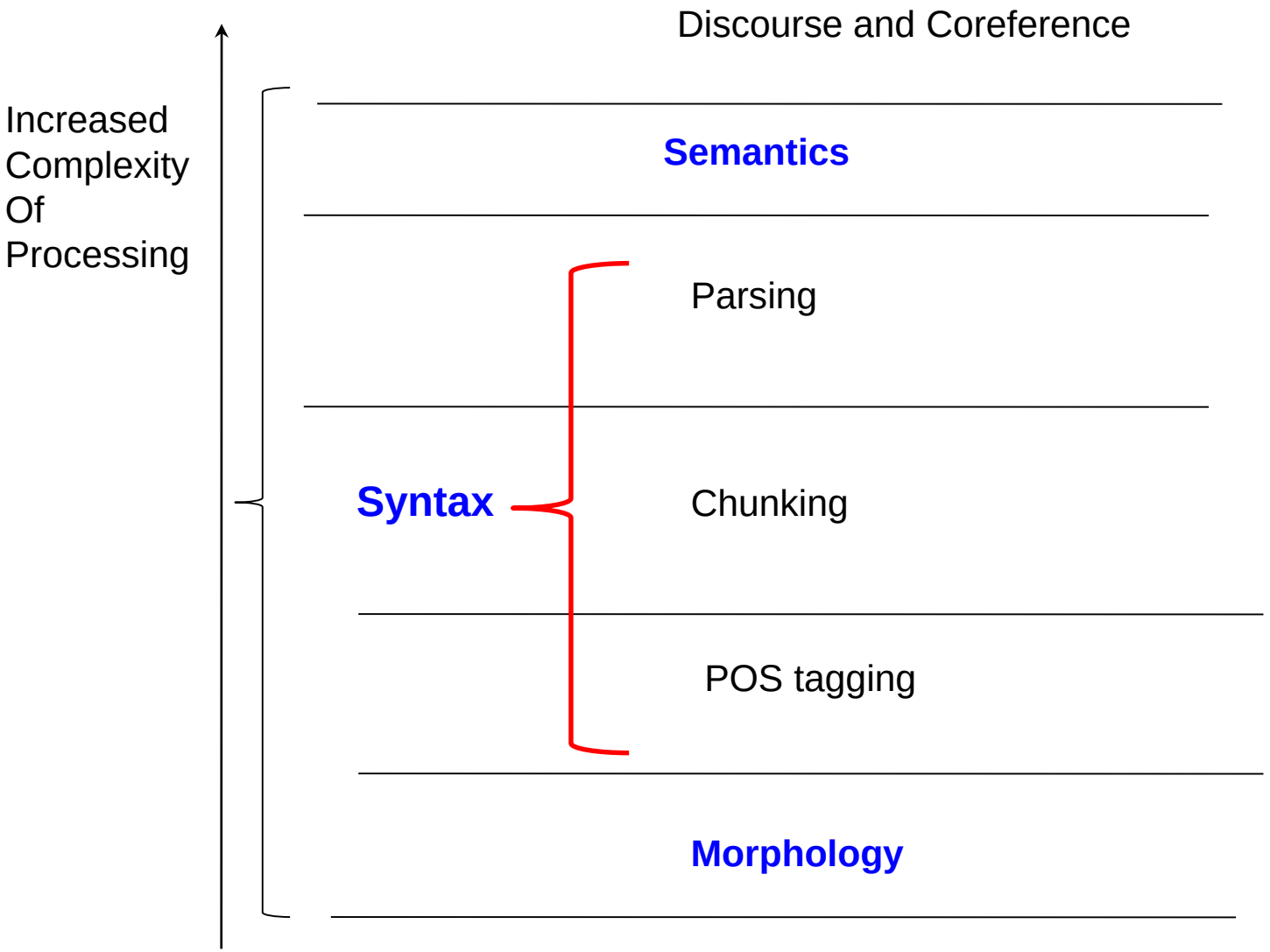
- *Posterior probability P(Cancer|patient_parameters),*
  - *Patient parameters: HBC count, weight, family history etc.*
- *Prior probability P(cancer)*
- *Likelihood P(patient_parameters| Cancer)*

Probabilities involved: visa

- *Posterior probability P(Grant_Visa|candidate_features)*
  - *Candidate features: income, education, travel history, etc.*
- *Prior probability P(Grant_Visa)*
- Likelihood *P(candidate_features|Grant_Visa)*

Part of Speech Tagging

NLP Layers


Discourse and Coreference
Increased
Complexity
Of
Processing
Semantics
Parsing
Syntax
Chunking
POS tagging
Morphology

argmax computation

$$T^* = \text{argmax}_T[P(T|W)]$$
$$= \text{argmax}_T = [\{P(T).P(W|T)\}/P(W)]$$
$$= \text{argmax}_T [\{P(T).P(W|T)\}]$$
$$= \text{argmax}_T [P(T,W)]$$

- Choose that *T* (called *T**) which has the highest probability given *W*
- Computation with *P(T|W)* is called **Discriminative Modelling**
- Computation with *P(T,W)* is called **Generative Modelling**

Argmax computation (1/2)

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

$$P(T) = P(t_0 = \hat{}\ t_1 t_2 \ldots t_{n+1} = .)$$
$$= P(t_0)P(t_1|t_0)P(t_2|t_1 t_0)P(t_3|t_2 t_1 t_0) \ldots$$
$$P(t_n|t_{n-1} t_{n-2} \ldots t_0)P(t_{n+1}|t_n t_{n-1} \ldots t_0)$$
$$= P(t_0)P(t_1|t_0)P(t_2|t_1) \ldots P(t_n|t_{n-1})P(t_{n+1}|t_n)$$
$$= \prod_{i=0}^{N+1} P(t_i|t_{i-1})$$

Bigram Assumption

Argmax computation (2/2)

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

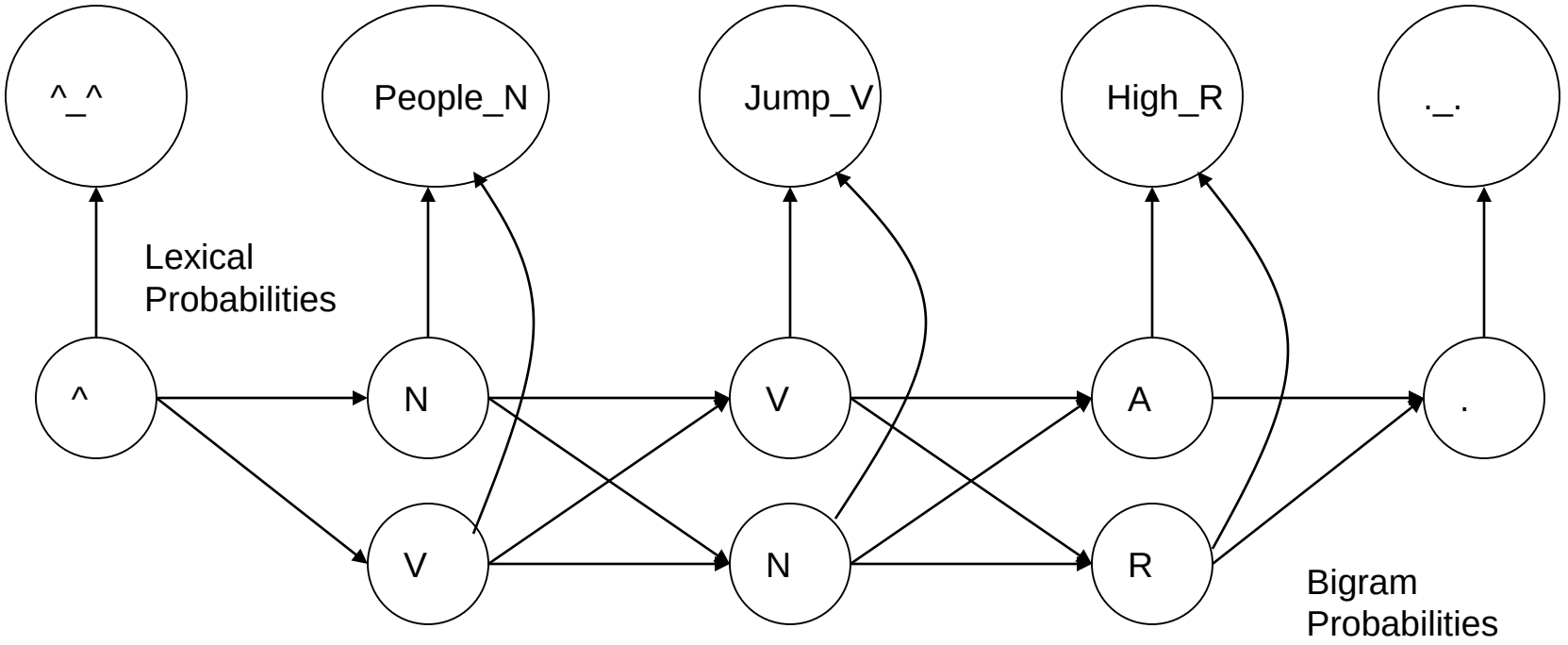
Assumption: A word is determined completely by its tag. This is inspired by speech recognition

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

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

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

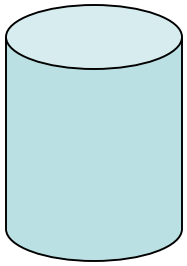
Generative Model



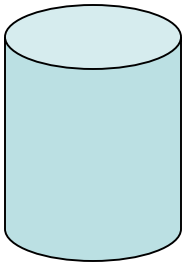
This model is called Generative model.
Here words are observed from tags as states.
This is similar to HMM.

An Explanatory Example

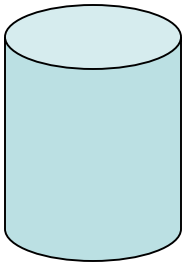
Colored Ball choosing



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

Probability of transition to another Urn after picking a ball:

	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

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

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

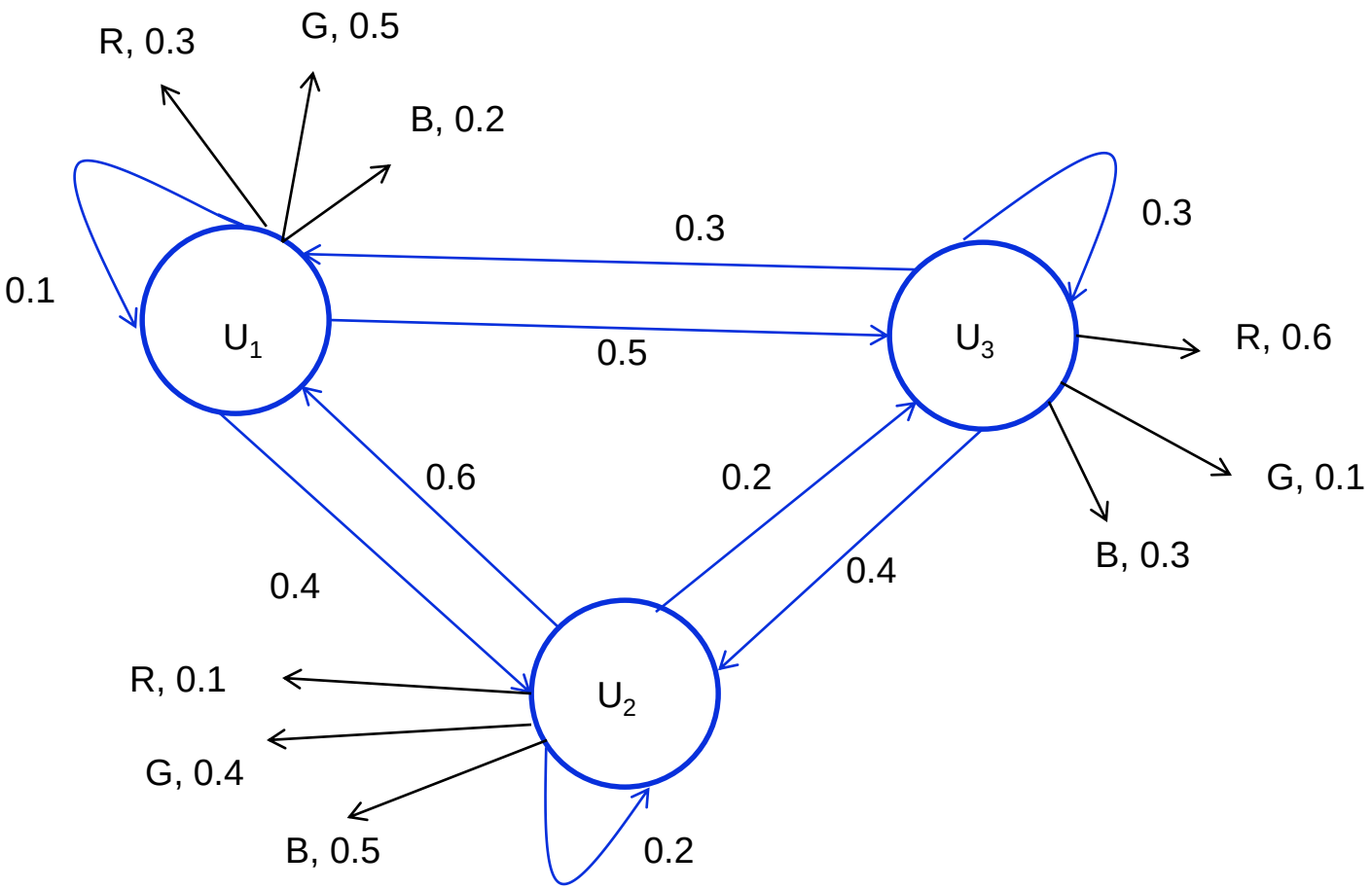
Observation : RRGGBRGR

State Sequence : ??

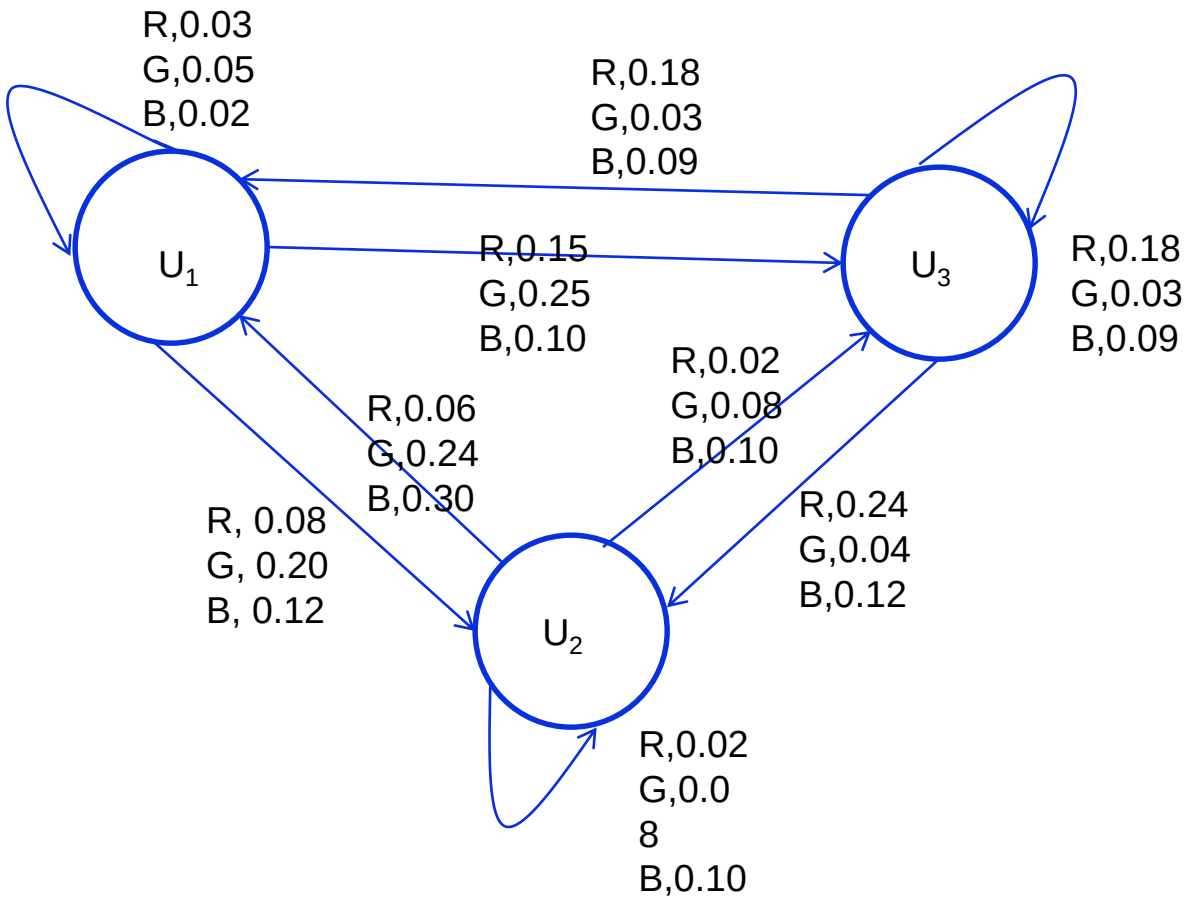
Not so Easily Computable.

There are also initial probabilities of starting A particular urn: 3 probabilities

Diagrammatic representation (1/2)



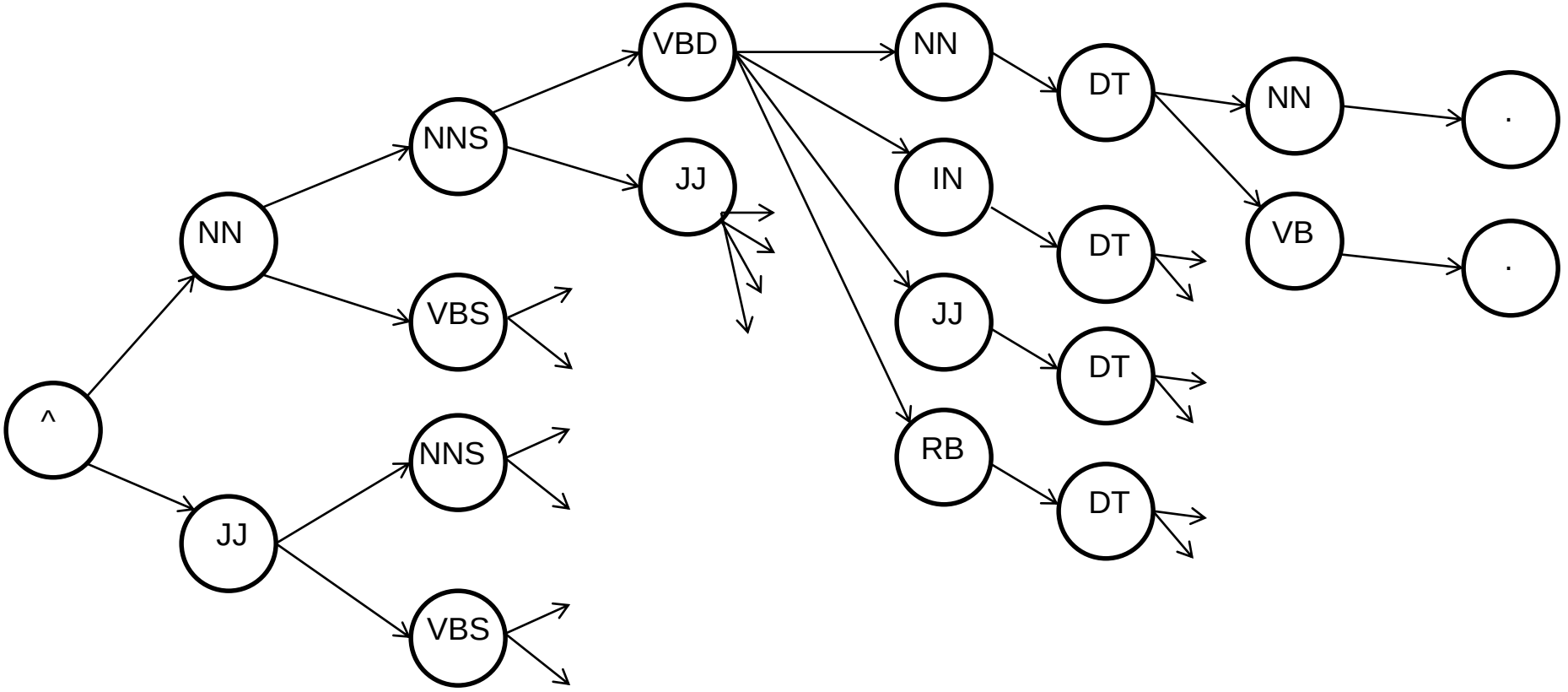
Diagrammatic representation (2/2)



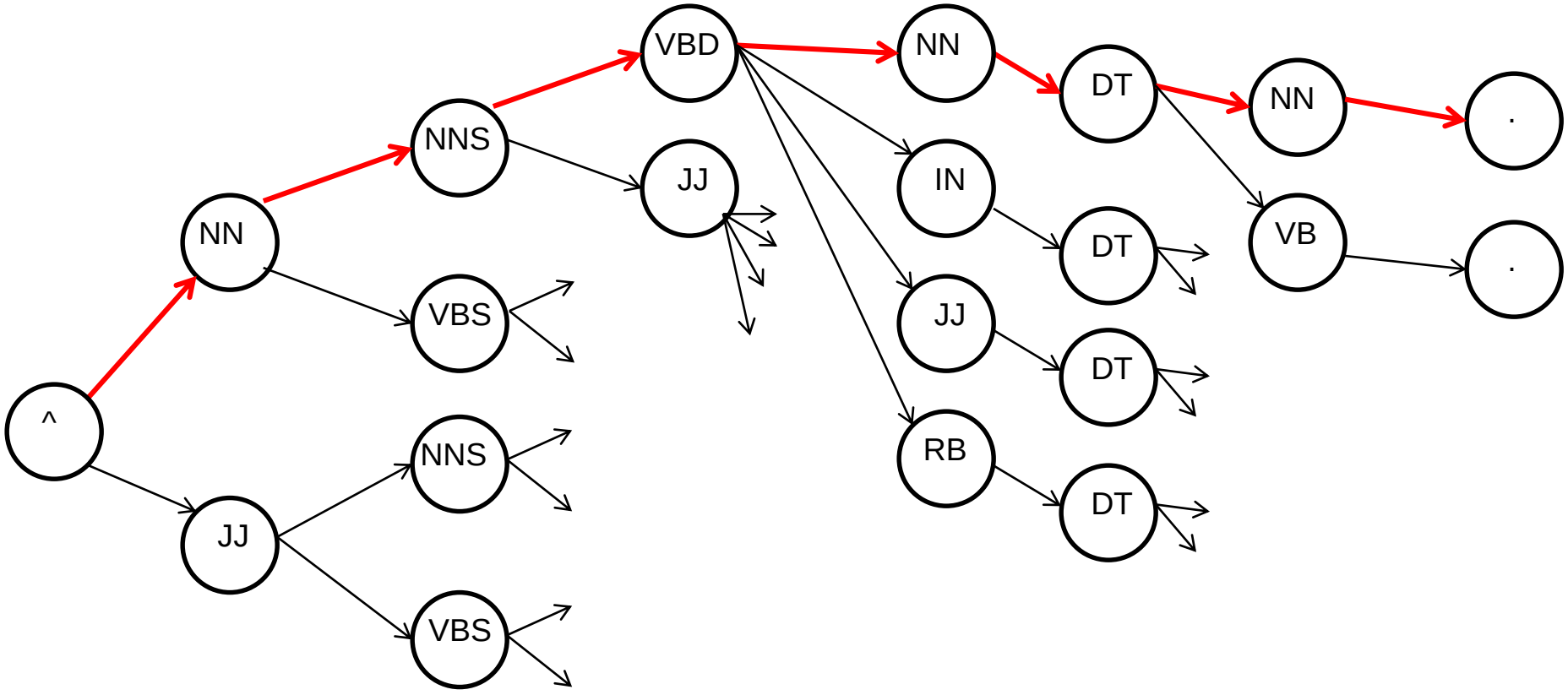
Computation of POS tags

DECODING

W:	^	Brown	foxes	jumped	over	the	fence	.
T:	^	JJ	NNS	VBD	NN	DT	NN	.
		NN	VBS	JJ	IN		VB	
					JJ			
					RB			



^ Brown foxes jumped over the fence .



^ Brown foxes jumped over the fence .

Probability of a path (e.g. Top most path) = *P*(T) * *P*(W|T)

P(^) . P(NN|^) . P(NNS|NN) . P(VBD|NNS) . P(NN|VBD) . P(DT|NN) . P(NN|DT) . P(.|NN) . P(.)

*

P(^|^) . P(brown|NN) . P(foxes|NNS) . P(jumped|VBD) . P(over|NN) . P(the|DT) . P(fence|NN) . P(.|.)

Illustration of Viterbi Decoding for POS tagging

From the book:

Pushpak Bhattacharyya and Aditya Joshi, *Natural Language Processing,* Wiley Eastern, 2023.

Sentence: "*People Dance*"

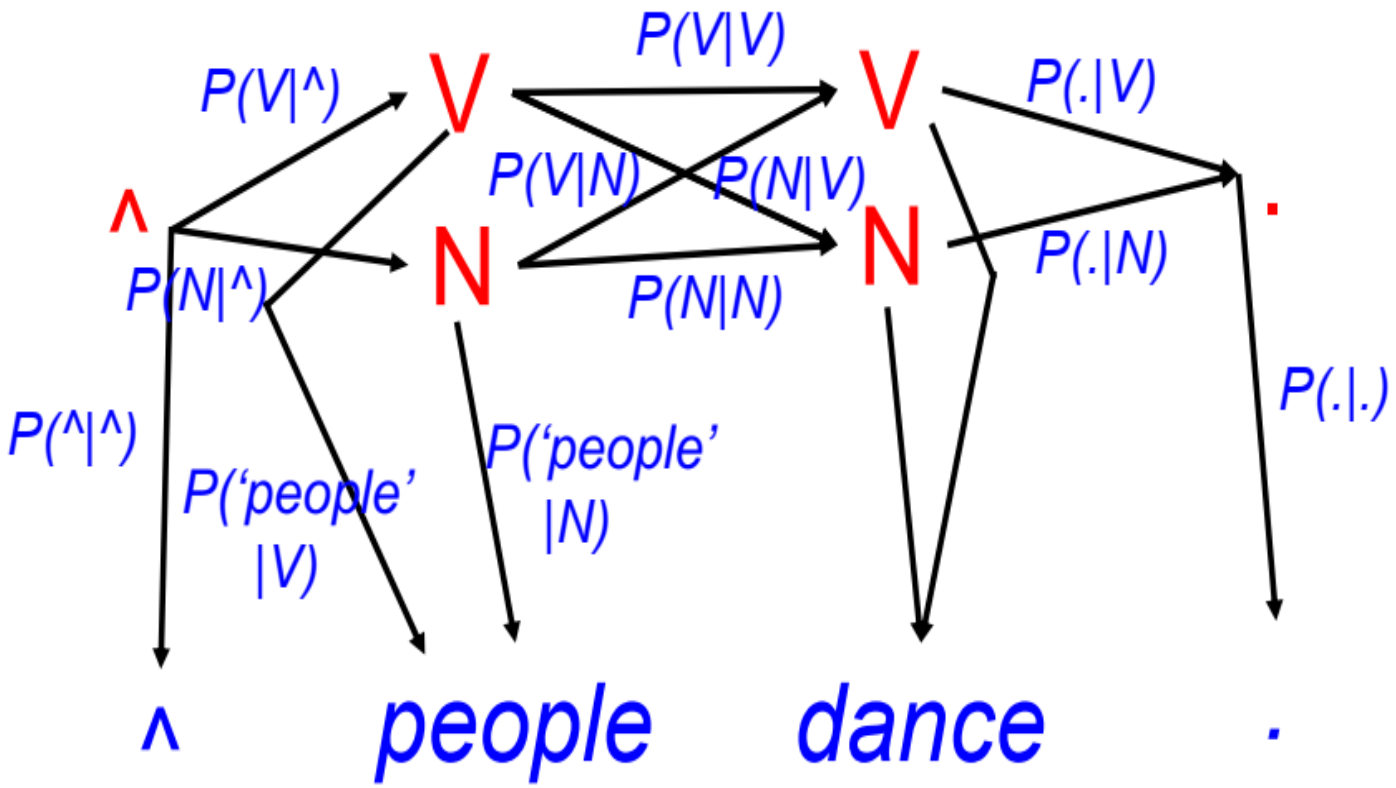
- '*people*' and '*dance*' can both be both nouns and verbs, as in
  - "*old_JJ people_NNS*" ('*people*' as noun)
  - "*township_NN peopled_VBN with soldiers_NNS*" ('*people*' as verb)
- as well as
  - "*rules_NNS of_IN classical_JJ dance_NN*" ('*dance*' as noun)
  - "*will*_VAUX *dance_VB well_RB*" ('*dance*' as verb)

Possible Tags: "^ *people dance .*"

- for simplicity we take single letter tags- *N: noun, V: verb*:
  - ^ *N N .*
  - ^ *N V .*
  - ^ *V N .*
  - ^ *V V .*
- We know that out of these, the second option ^ *N V.* is the correct one. How do we get this sequence?

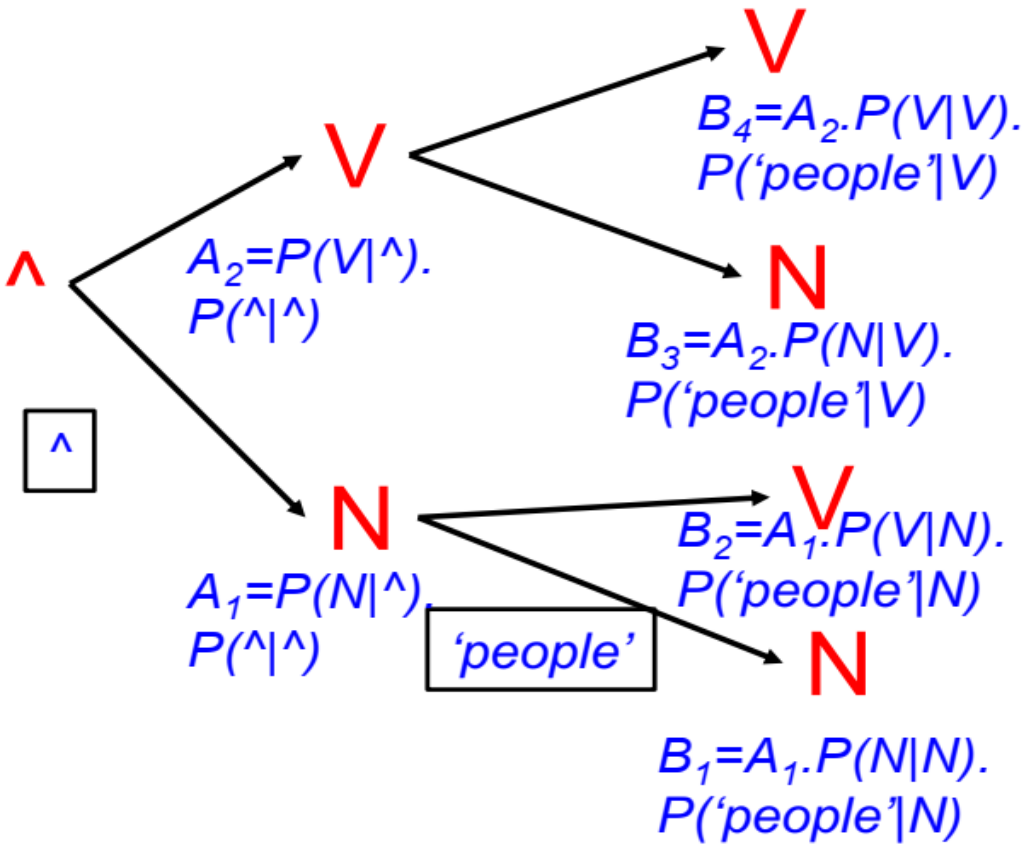
Step-1: Trellis

Columns of tags on each input word with transition arcs going from tags (states) to tags in consecutive columns and output arcs going from tags to words (observations)



Aim: select the highest probability path

From 4 possibilities; *A*s and *B*s are accumulated probabilities



Some numerical values: hypothetical but not unrealistic

- Calculations:
- When it comes to the start of the sentence, most sentences start with a noun. So lets have

  $P(N|\hat{})=0.8$, $P(V|\hat{})=0.2$ and of course $P(\text{“}\hat{}\text{”}|\hat{})=1.0$
- Then

  $A_1=0.8$, $A_2=0.2$

Encounter “people”: more probabilities (1/2)

- Transition from *N* to *N* is less common than to *V*.
- Transition from *V* to *V*- as in auxiliary verb to main verb- is quite common (e.g., *is going*).
- *V* to *N* too is common- as in case of a nominal object following the verb *(going home).*
- Following plausible transition probabilities:
  - *P(N|N)=0.2, P(V|N)=0.8, P(V|V)=0.4, P(N|V)=0.6*
- We also need lexical probabilities. ‘people’ appearing as verb is much less common than its appearing as noun. So let us have

Encounter “people”: more probabilities (2/2)

- We also need lexical probabilities. ‘people’ appearing as verb is much less common than its appearing as noun. So let us have

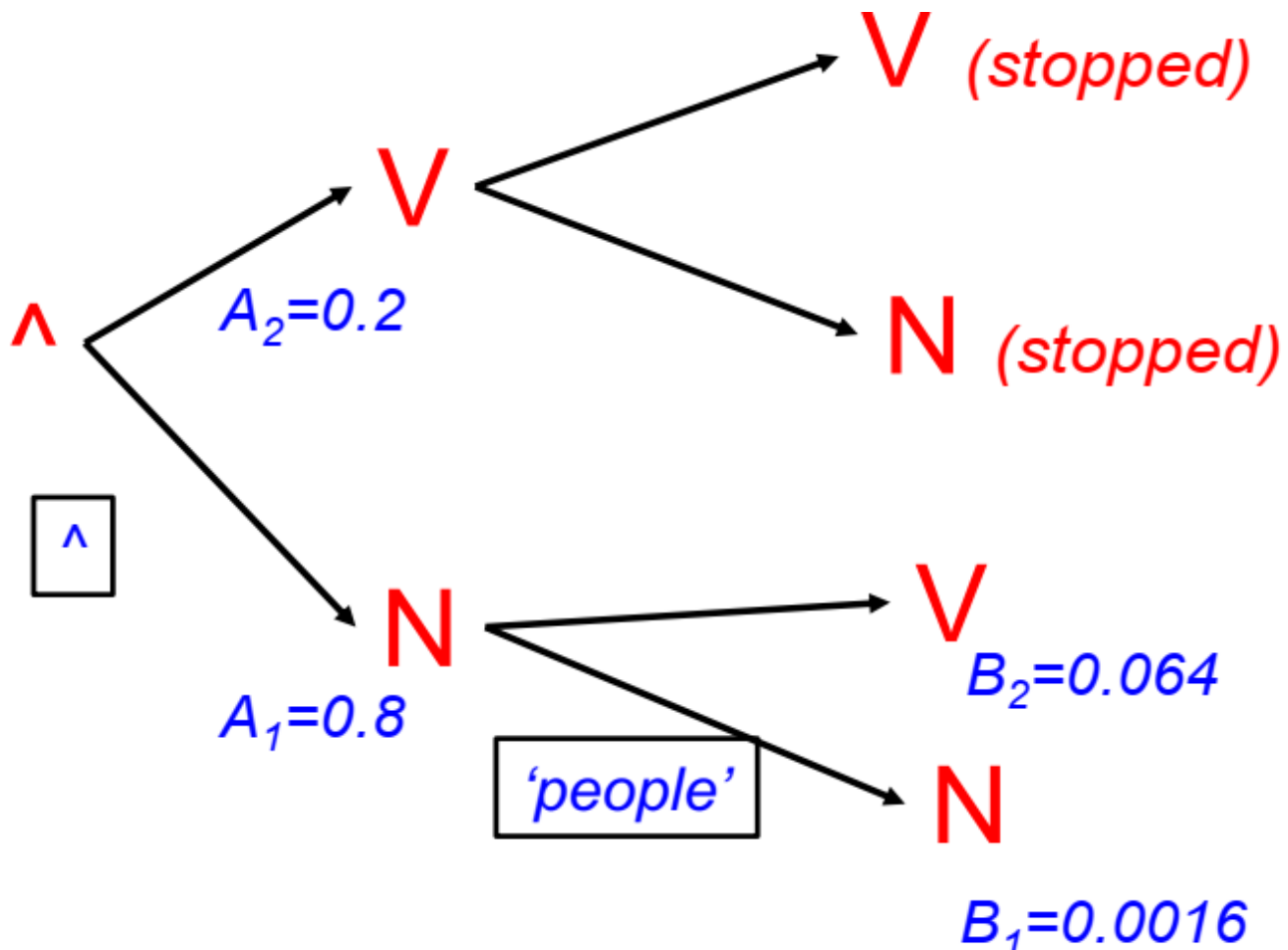
  - *P(‘people’|N)=0.01, P(‘people’|V)=0.001*
- Note: *N N*: golf club, cricket bat, town people- ambiguity “The town people visited was deserted”/”town people will not be able to live here”
- V V combination: Hindi- *has padaa* (laughed suddenly), Bengali- *chole gelo* (went away)

Calculate *B*s

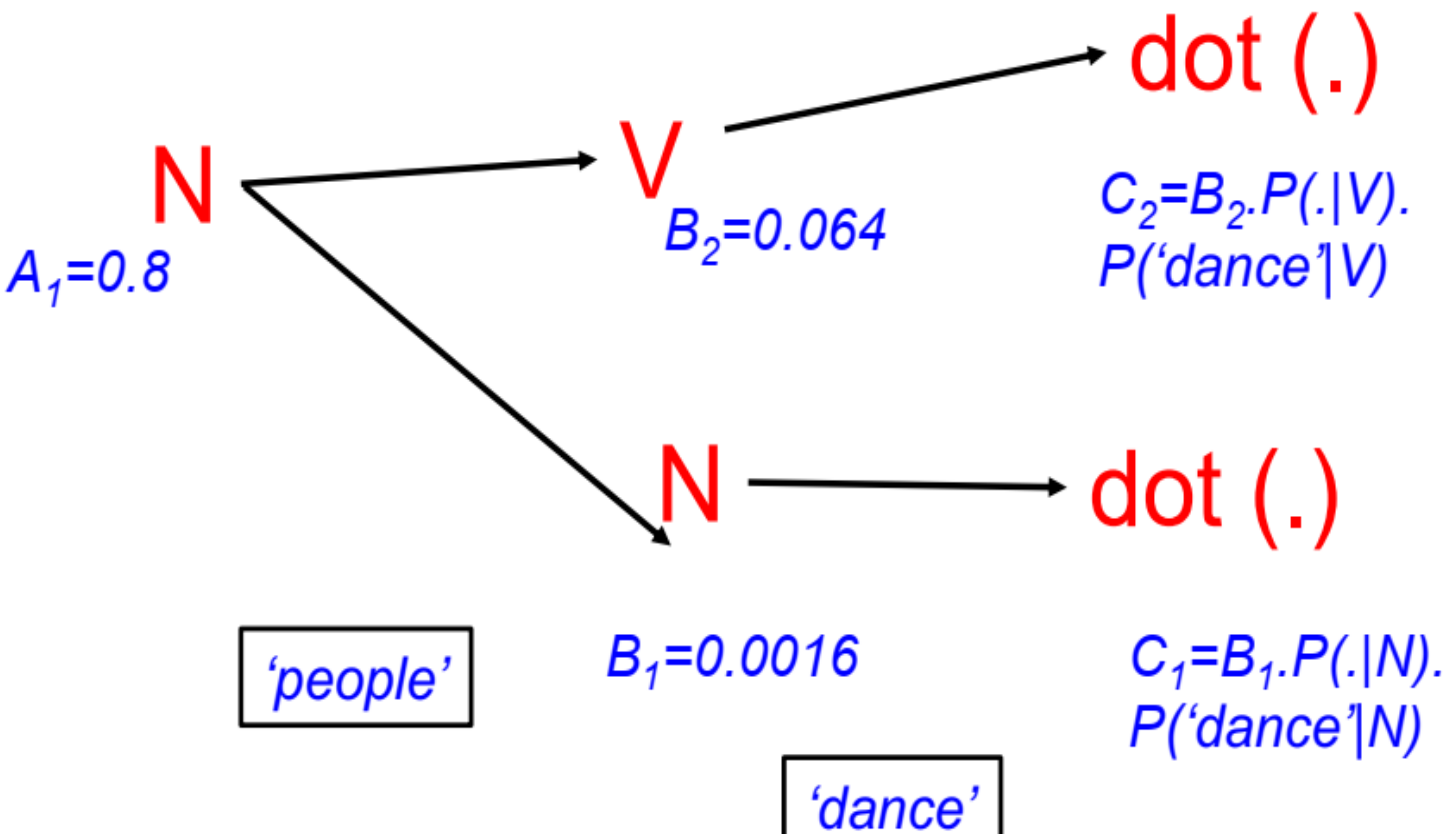
- B_1=0.8.0.2.0.01=0.0016 (approx.)
- B_2=0.8.0.8.0.01=0.064 (approx.)
- B_3=0.2.0.6.0.001=0.00012
- B_4=0.2.0.4.0.001=0.00008

Reduced Viterbi Configuration

- Heart of Decoding → linear time



Next word: '*dance*'



More probabilities needed

- We can give equal probabilities to sentences ending in noun and verb. Also, ‘dance’ as verb is more common than noun.

$$P(.|N)=0.5=P(.|V)$$

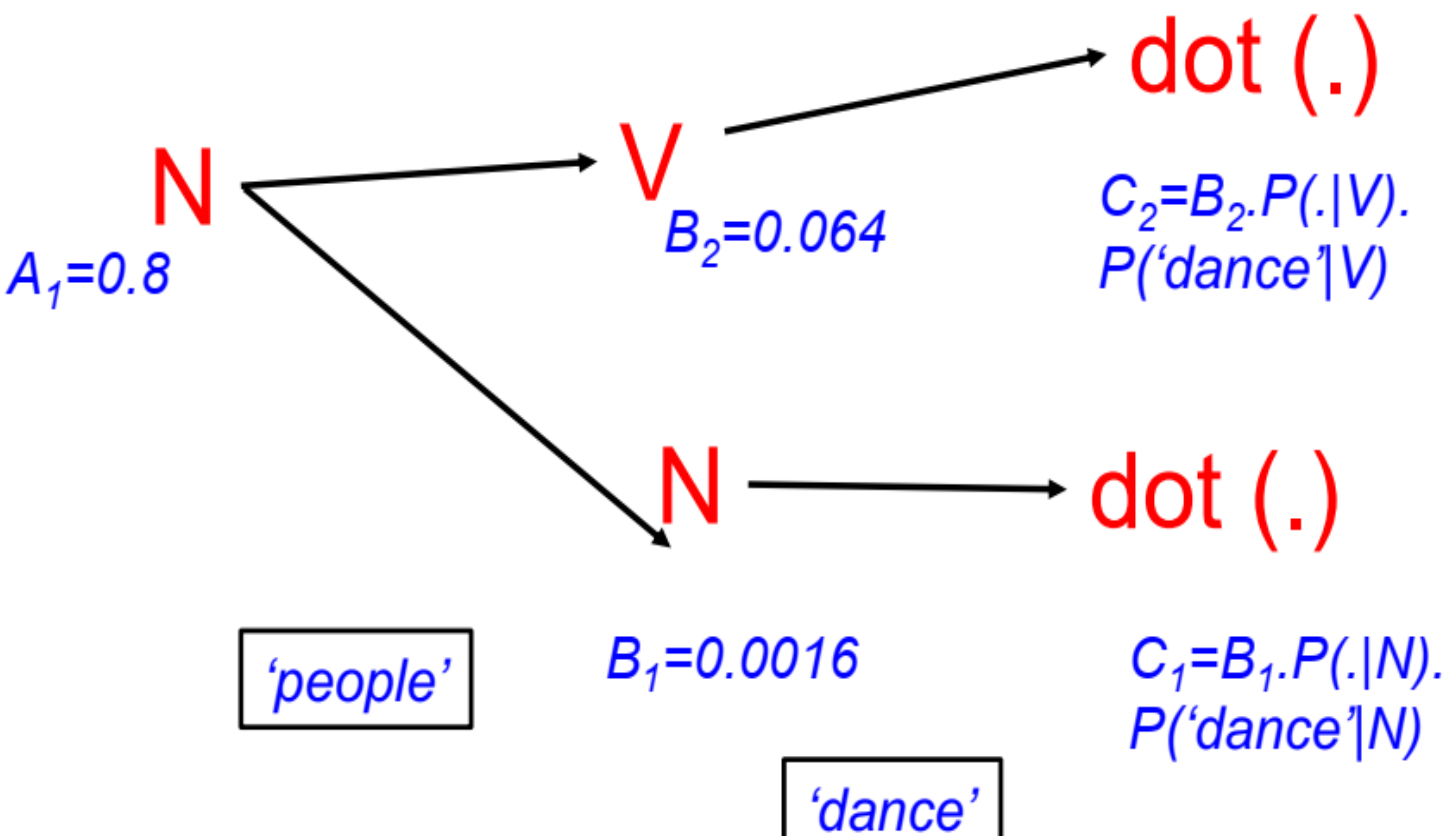
$$P(\text{‘dance’}|N)=0.001$$

$$P(\text{‘dance’}|V)=0.01$$

Best Path: ^ N V .

C_1=0.0016.0.5.0.001=0.0000008

C_2=0.064.0.5.0.01=0.00032



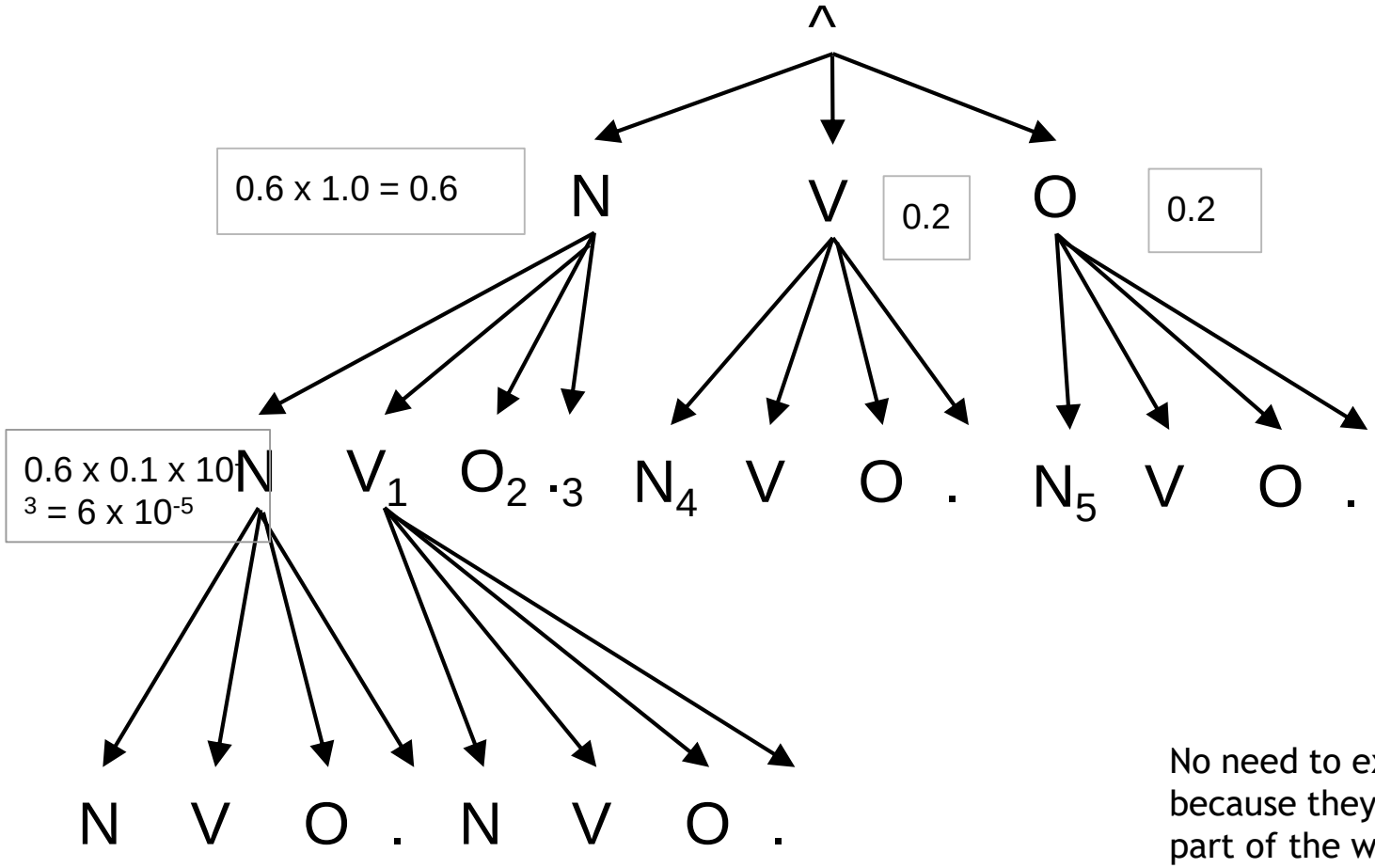
Computational Complexity

- 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



ϵ

people

laugh

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

[1] $0.6 \times 0.4 \times 10^{-3} = 2.4 \times 10^{-4}$

[2] $0.6 \times 0.3 \times 10^{-3} = 1.8 \times 10^{-4}$

[3] $0.6 \times 0.2 \times 10^{-3} = 1.2 \times 10^{-4}$

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.

Problems faced in HMM based POS tagging

All problems are due to SPARSITY; following are the kinds of sparsity

(a) unseen words (‘delay’ in training corpus, but not ‘procrastination’)

(b) Different form of the word- morphology (corpus has ‘predictable’, but not ‘unpredictable’)

(c) has code mixing (‘aap mujhe advice dijiye’- ‘you give me advice)

(d) The corpus does not have the particular word-tag combination (‘people’ as verb)

Three basic Tasks wrt HMM

- 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)

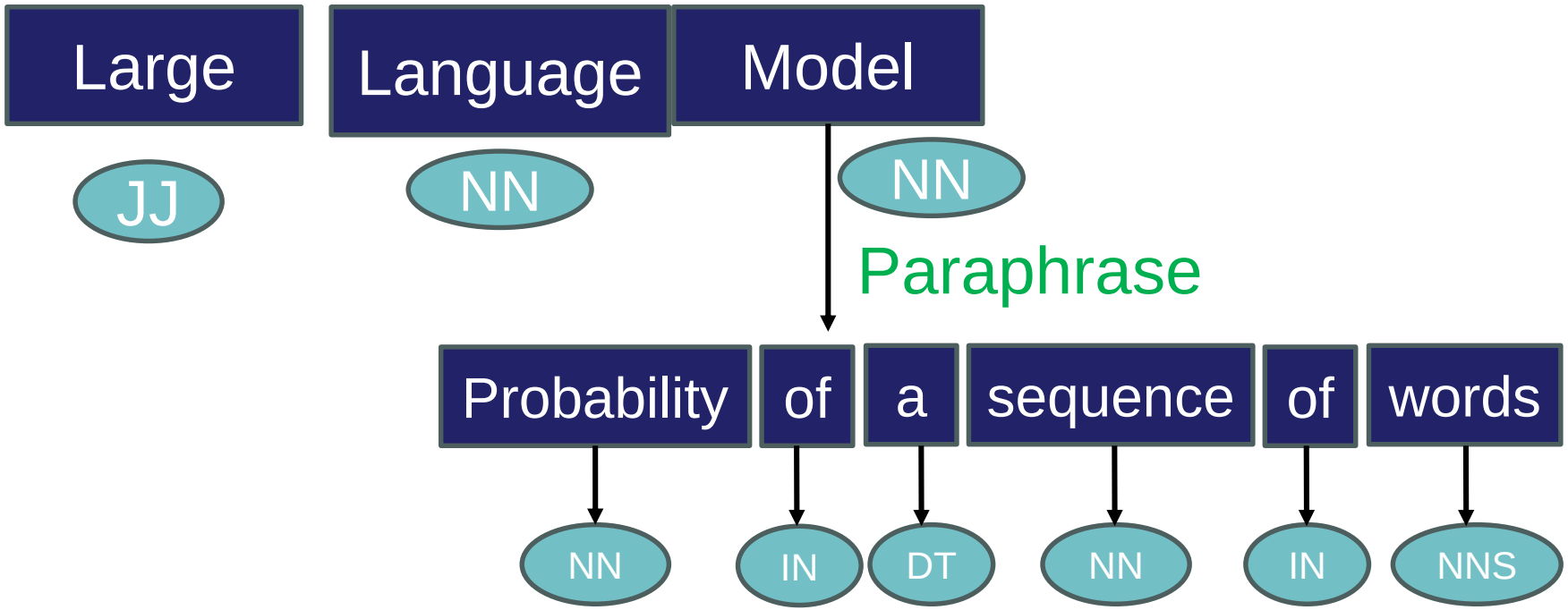
A note on Problem 1

Probability of observation

- Problem 1: Likelihood of a sequence
  - Forward Procedure
  - Backward Procedure
- If the observation is a word sequence, it is called “Language Model (LM)”

- In that case, what does LLM mean?

Meaning of “Large Language Model”



Meaning of LLM: Probability of a “large” (long) sequence of words; GPT4→ context is 4000 words

Definition of Prob. of a **sequence of words**

Probability of a **sequence of words**

≡

Probability of **occurrence in corpus**