POS Tagging and Chunking

By

Manish Shrivastava

POS Tagging Basics

- Assign “Part of Speech” to each word in a given sentence
- Example:

- This/DT is/VBZ a/DT sentence/NN ./.
- This/DT is/VBZ a/DT tagged/JJ sentence/NN ./.

Where does POS tagging fit in

Increased
Complexity
Of
Processing

Discourse and Corefernce

Semantics Extraction

Parsing

Chunking

POS tagging

Morphology

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***

Simple Method

- Assign the most common tag
- Example :
  - I/PRP bank/NN at/IN SBI/NNP ./SYM
- But, the correct tag sequence in context is:
  - I/PRP bank/VBP at/IN SBI/NNP ./SYM
- Assign "Part of Speech" to each word according to its context in a given sentence

Mathematics of POS Tagging

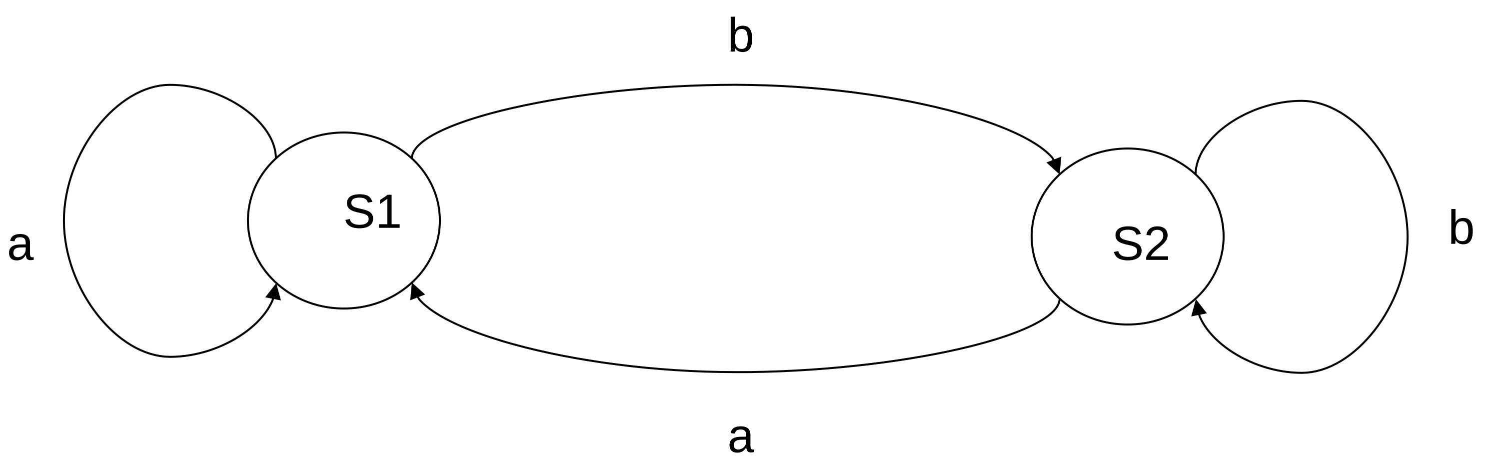
- Formally,
  - POS Tagging is a sequence labeling task
- For a given observation sequence W
  - W: $\{w_1, w_2 \ldots w_n\}$
- Produce a label/tag sequence T
  - T: $\{t_1, t_2 \ldots t_n\}$
- Such that they "belong" together
  - Maximize P(W,T) or P(T|W)
  - argmax P(T|W)

Computing Label sequence given Observation Sequence

- At first glance, It seems straight forward to directly compute/learn P(T|W)
  - Any Suggestions…??
-

- It is not possible to directly compute P(T|W) from the data
- So, we use Bayes' Theorem
  - P(T|W) = P(T)P(W|T)/P(W)
  - maximizing this term, we get
  - T*= $\underset{T}{\operatorname{argmax}} P(T|W) = \underset{T}{\operatorname{argmax}} P(W|T)P(T)$

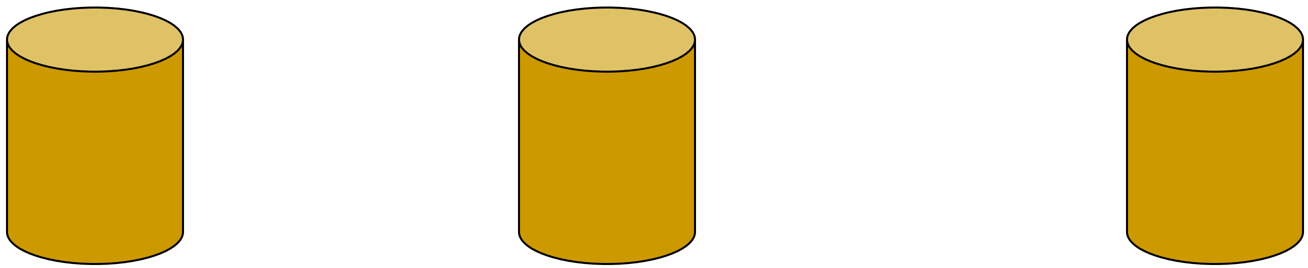
A Simple Process



A simple automata

A Slightly Complicated Process

A colored ball choosing example :



<u>Urn 1</u>
# of Red = 100
# of Green = 0
# of Blue = 0

<u>Urn 2</u>
# of Red = 0
# of Green = 100
# of Blue = 0

<u>Urn 3</u>
# of Red = 0
# of Green = 0
# of Blue = 100

Probability of transition to another Urn after picking a ball:

	U1	U2	U3
U1	0.1	0.4	0.5
U2	0.6	0.2	0.2
U3	0.3	0.4	0.3

A Slightly Complicated Process contd.

Given :

	U1	U2	U3
U1	0.1	0.4	0.5
U2	0.6	0.2	0.2
U3	0.3	0.4	0.3

Observation : RRGGBRGR

State Sequence : ??

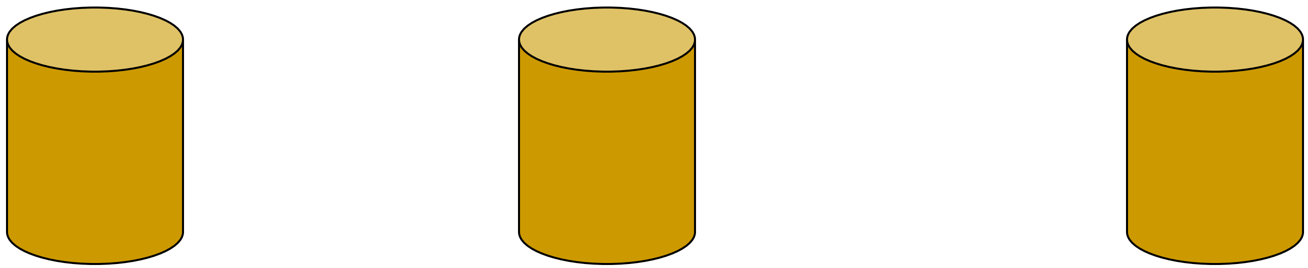
Easily Computable.

Markov Processes

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

A (Slightly Complicated) Markov Process

A colored ball choosing example :



Urn 1
# of Red = 100
# of Green = 0
# of Blue = 0

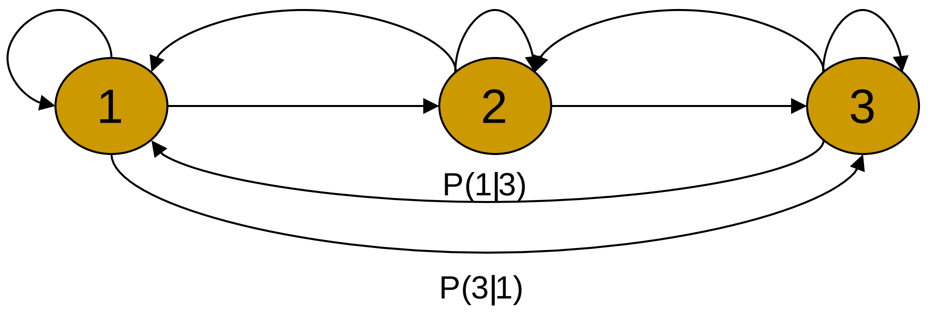
Urn 2
# of Red = 0
# of Green = 100
# of Blue = 0

Urn 3
# of Red = 0
# of Green = 0
# of Blue = 100

Probability of transition to another Urn after picking a ball:

	U1	U2	U3
U1	0.1	0.4	0.5
U2	0.6	0.2	0.2
U3	0.3	0.4	0.3

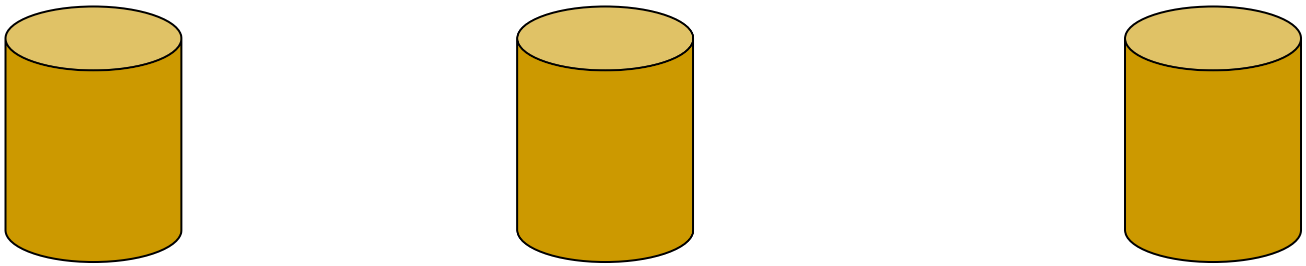
Markov Process



- Visible Markov Model
  - Given the observation, one can easily follow the state sequence traversed

Hidden Markov Model

A colored ball choosing example :



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:

	U1	U2	U3
U1	0.1	0.4	0.5
U2	0.6	0.2	0.2
U3	0.3	0.4	0.3

Hidden Markov Model

Given :

	U1	U2	U3
U1	0.1	0.4	0.5
U2	0.6	0.2	0.2
U3	0.3	0.4	0.3

and

	R	G	B
U1	0.3	0.5	0.2
U2	0.1	0.4	0.5
U3	0.6	0.1	0.3

Observation : RRGGBRGR

State Sequence : ??

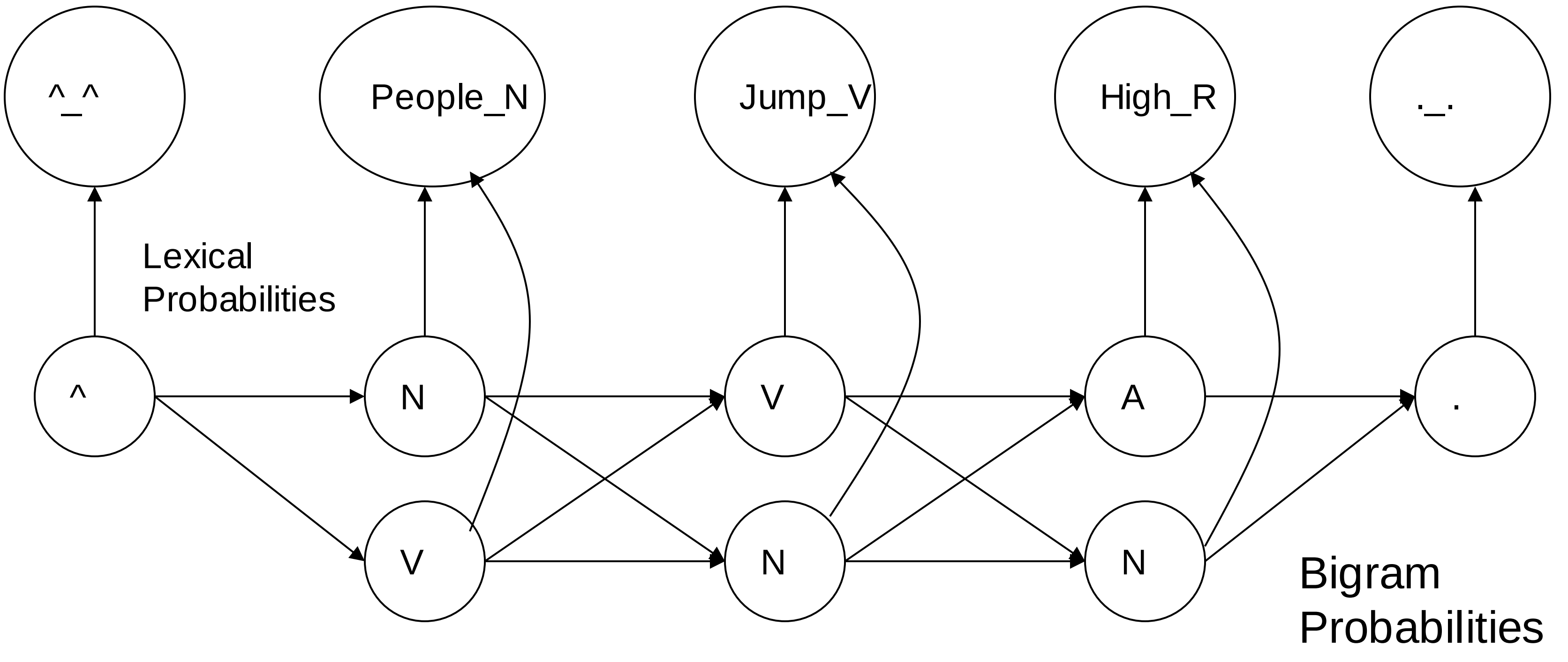
Not So Easily Computable.

Hidden Markov Model

- Set of states : S
- Output Alphabet : V
- Transition Probabilities : $A = \{a_{ij}\}$
- Emission Probabilities : $B = \{b_j(o_k)\}$
- Initial State Probabilities : π

$$\lambda = (A, B, \pi)$$

Generative Model



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

Hidden Markov Model

- 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 =

	U1	U2	U3
U1	0.1	0.4	0.5
U2	0.6	0.2	0.2
U3	0.3	0.4	0.3

B=

	R	G	B
U1	0.3	0.5	0.2
U2	0.1	0.4	0.5
U3	0.6	0.1	0.3

Three Basic Problems of HMM

1. Given Observation Sequence $O = \{o_1 \ldots o_n\}$
   - Efficiently estimate $P(O \mid \lambda)$
2. Given Observation Sequence $O = \{o_1 \ldots o_n\}$
   - Get best $Q = \{q_1 \ldots q_n\}$
3. How to adjust $\lambda = (A, B, \pi)$ to best maximize $P(O \mid \lambda)$

Solutions

- 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)

Problem 1

$$\text{Consider}: \mathrm{O} = \{\mathrm{o}_1 \ldots \mathrm{o}_\mathrm{T}\}$$

$$\text{And } \mathrm{Q} = \{\mathrm{q}_1 \ldots \mathrm{q}_\mathrm{T}\}$$

Then ,

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda)$$

$$= b_{q_1}(o_1). \ \ldots b_{q_T}(o_T)$$

And

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} \ldots . a_{q_{T-1} q_T}$$

We know,

$$P(O \mid \lambda) = \sum_{T} P(O, Q \mid \lambda)$$

Then,

$$P(O \mid \lambda) = \sum_{T} P(O \mid Q, \lambda) P(Q \mid \lambda)$$

Problem 1

$$P(O \mid \lambda) = \sum_{q_1 \ldots q_T} \pi_{q_1} a_{q_1 q_2} \ldots . a_{q_{T-1} q_T} b_{q_1}(o_1) . \ldots b_{q_T}(o_T)$$

- Order $2TN^T$
- Definitely not efficient!!
- Is there a method to tackle this problem? Yes.
  - Forward or Backward Procedure

Forward Procedure

Define Forward variable as

$\alpha_t(i) = P(o_1 \dots o_t, q_t = S_i \mid \lambda)$

The probability that the state at position t is S_i,

and of the partial observation $o_1 \dots o_t$, given the model λ

Forward Step:

1) **Initialization:**

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N.$$

2) **Induction:**

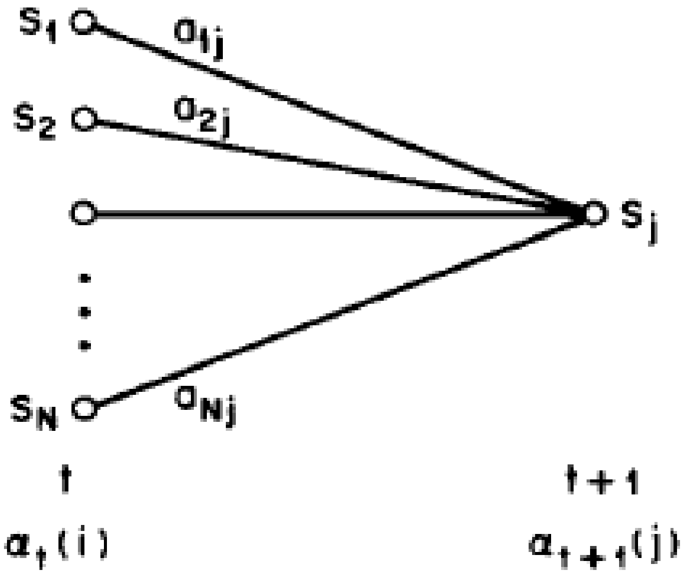
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \qquad 1 \le t \le T-1$$

$$1 \le j \le N.$$

3) **Termination:**

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

Forward Procedure



Backward Procedure

$$\beta_t(i) = P(O_{t+1}\, O_{t+2} \cdots O_T | q_t = S_i, \lambda)$$

1) Initialization:

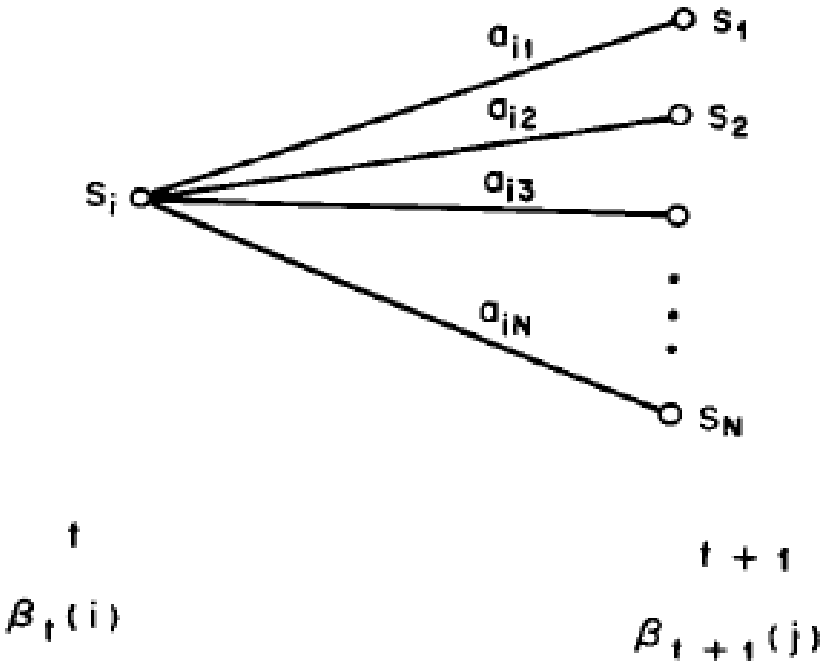
$$\beta_T(i) = 1, \qquad 1 \le i \le N.$$

2) Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j),$$

$$t = T-1, T-2, \cdots, 1, \quad 1 \le i \le N.$$

Backward Procedure



Forward Backward Procedure

- Benefit
  - Order
    - N^2T as compared to $2TN^T$ for simple computation
- Only Forward or Backward procedure needed for Problem 1

Problem 2

- Given Observation Sequence O ={o_1… o_T}
  - Get "best" Q ={q_1… q_T} i.e.
- Solution :
  1. Best state individually likely at a position i
  2. Best state given all the previously observed states and observations
     - Viterbi Algorithm

Viterbi Algorithm

• Define $\delta_t(i)$ such that,

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1\, q_2 \cdots q_t = i,\, O_1\, O_2 \cdots O_t | \lambda]$$

i.e. the sequence which has the best joint probability so far.

• By induction, we have,

$$\delta_{t+1}(j) = [\max_i \delta_t(i)\, a_{ij}] \cdot b_j(O_{t+1}).$$

Viterbi Algorithm

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N$$

$$\psi_1(i) = 0.$$

2) Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \qquad 2 \le t \le T$$

$$1 \le j \le N$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], \qquad 2 \le t \le T$$

$$1 \le j \le N.$$

Viterbi Algorithm

3) Termination:

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \underset{1 \le i \le N}{\operatorname{argmax}} [\delta_T(i)].$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \cdots, 1.$$

Problem 3

- How to adjust $\lambda = (A, B, \pi)$ to best maximize $P(O \mid \lambda)$
  - Re-estimate λ
- Solutions :
  - To re-estimate (iteratively update and improve) HMM parameters A,B, π
    - Use Baum-Welch algorithm

Baum-Welch Algorithm

- Define $\xi_t(i, j)$

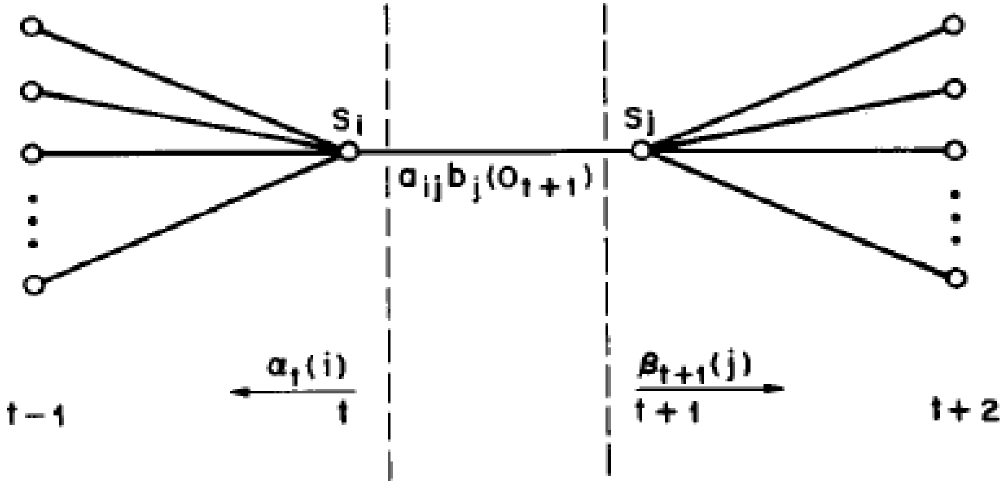
$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda).$$

- Putting forward and backward variables

$$\xi_t(i, j) = \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}$$

Baum-Welch algorithm



Baum-Welch algorithm

- Define $\gamma_t(i)$:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j).$$

- Then, expected number of transitions from S_i

$$\sum_{t=1}^{T-1} \gamma_t(i)$$

- And, expected number of transitions from S_j to S_i

$$\sum_{t=1}^{T-1} \xi_t(i, j)$$

Baum-Welch algorithm

$$\bar{\pi}_i = \text{expected frequency (number of times) in state } S_i \text{ at time } (t = 1) = \gamma_1(i)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$= \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.$$

Baum-Welch Algorithm

- Baum et al have proved that the above equations lead to a model as good or better than the previous

Shallow Parsing/Chunking

Goal: Divide a sentence into a sequence of chunks.

- Chunks are non-overlapping regions of a text

[I] saw [a tall man] in [the park].

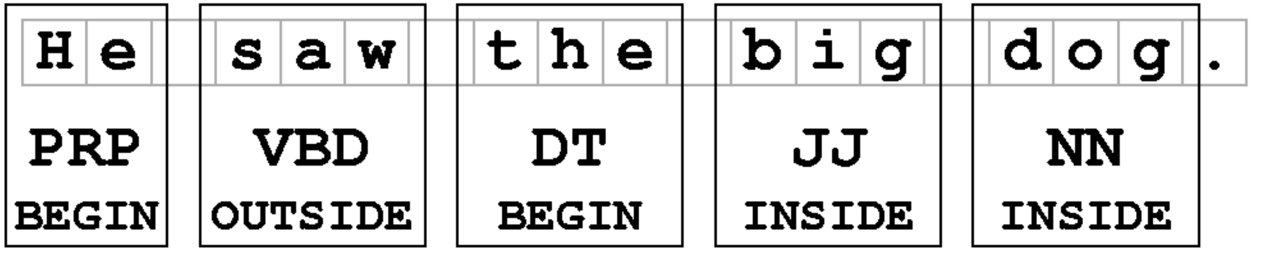
- Chunks are non-recursive
  - A chunk can not contain other chunks
- Chunks are non-exhaustive
  - Not all words are included in chunks

Motivation

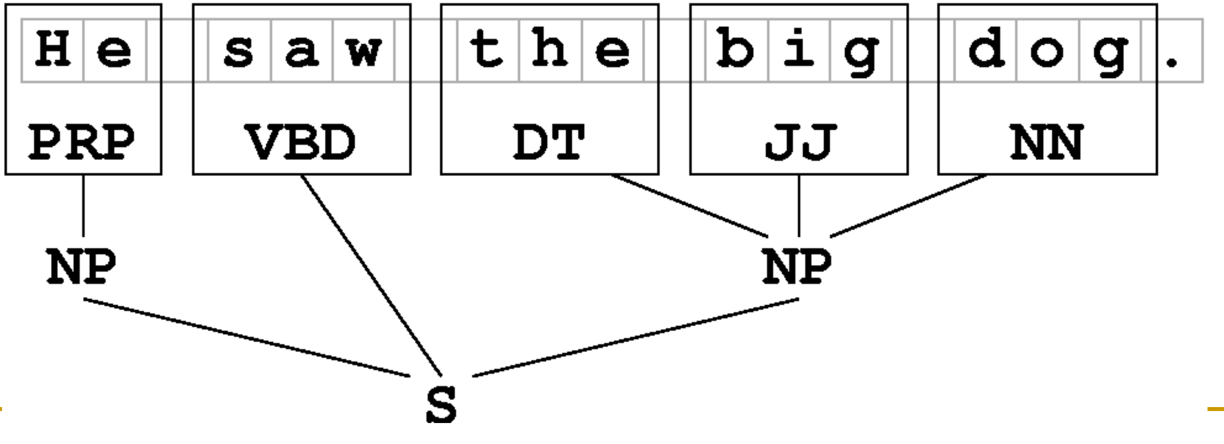
- Locating information
  - e.g., text retrieval
    - Index a document collection on its noun phrases
- Ignoring information
  - Generalize in order to study higher-level patterns
    - e.g. phrases involving “gave” in Penn treebank:
      - gave NP; gave up NP in NP; gave NP up; gave NP help; gave NP to NP

Representation

- BIO (or IOB)



Trees



Comparison with Full Parsing

- Shallow parsing is an easier problem
  - Less word-order flexibility within chunks than between chunks
  - More locality:
    - Fewer long-range dependencies
    - Less context-dependence
    - Less ambiguity

Chunks and Constituency

***Constituents:* [[a tall man] [in [the park]]].**

***Chunks:* [a tall man] in [the park].**

- A constituent is part of some higher unit in the hierarchical syntactic parse
- Chunks are *not constituents*
  - Constituents are recursive
- But, chunks are typically subsequences of constituents
  - Chunks do not cross major constituent boundaries

Statistical Methods for Chunking

- HMM
  - BIO tags
    - B-NP, I-NP, O
  - Sequence Labelling task

References

- Lawrence R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77 (2), p. 257–286, February 1989.
- Chris Manning and Hinrich Schütze, Chapter 9: Markov Models, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999