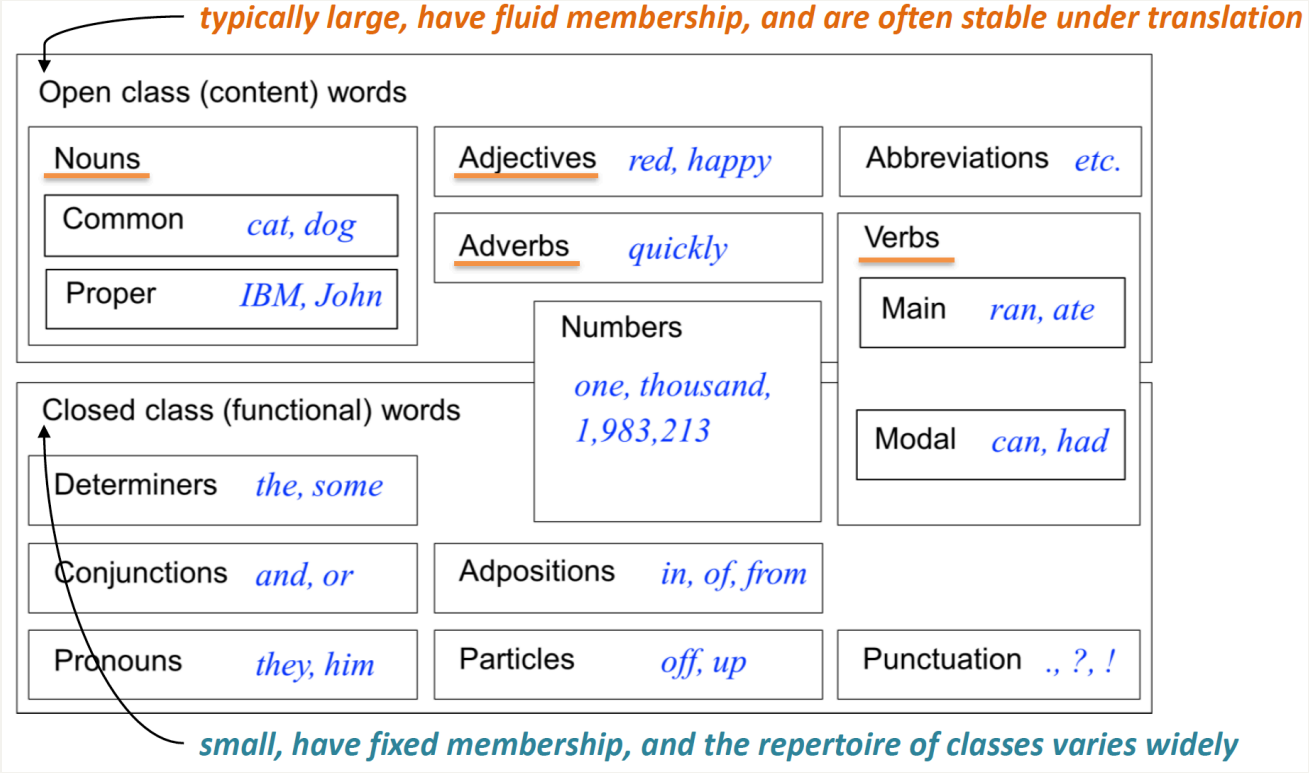
Lecture 6 POS Tagging

POS overview

Note 1. In English Tagset, larger and mode fine-grained tagsets are preferred

Note 2. lingustic structure: syntactic word classes



Criteria for POS tagging

- Distrbutional

  — where can the words occur?

  — e.g. nouns can appear with possession, like "her car","his car"

  — e.g. different types of verbs have different distributional properties, "to jump","he jumps"

- Morphological

  — what for does the word have?

  — what affixes can it take?

  — e.g ness-, -tion,-ity, and -ance tend to indicate nouns

  — e.g. -ate, -ize tend to be verbs; -ing tend to be the present participle
- Notional(or Semantic)

  — what sort of concept does the word refer to?

  — e.g nouns generally refer to living things, places, non-living thighs, or concepts

  —e.g. verbs refer to actions

Purpose of POS tagging

- Useful in and of itself

  — TTS

  — Lemmatization

  — Linguistically motivation word clustering

- Useful as a pre-processing step for parsing
- Useful as features to downstream systems

Issue of Pos tagging

- Ambiguous to identify the correct POS for word with more than one

Baseline Approaches

Lexicon-based POS Tagging

- Look up the word in the dictionary and look up the POS tag

- Annotate word with tag without considering informatino
- Indentify words present on lexicon
- Aggregate result
- cannot handle unknown words

Rule-based POS Tagging

- Basic idea (old POS taggers):

  — identify possible POS tags for each word in the sentence based on lexicon.

  — use set of hand-craft rules to select a POS from each tag list for each word.

- Approach

  — Assign each token all their possible tags

  — Apply rules that eliminate all tags that are in consistent with its context

  — Assign unknow word tokens a tag (the most frequent tag) that is consistent with its context

- Disadvantages:

  — used a large set of hand-crafted context-sensitive rules

  — cannot eliminate all POS ambiguity

N-gram with POS tagging

- Approaches:

  — take n-gram tokens as the input

  — calculate the frequency matrix of the POS given the POS of the preceding word

  — given a new text, tag the words from left to right with the most likely tag given the preceding one

- Problems:

  — Frequency matrix, will quicly get very large and sparse

  — One incorrect tagging choice might have unintended effects

  — No lookahead: choosing the "most probable" tag at one stage,

  might lead to highly imporbalble choice later.

  — As we are looking for the overall most likely tagging sequence given the bigram frequencies

Probabilistic Approaches

Hidden Markiv Model

Markove Chain

- A stochastic model used to model randomly changing system to

  — assumes the markov property

- A stochastic process has the Markov property, if

  — conditional propability of future state of the process

  — depends only on the present state

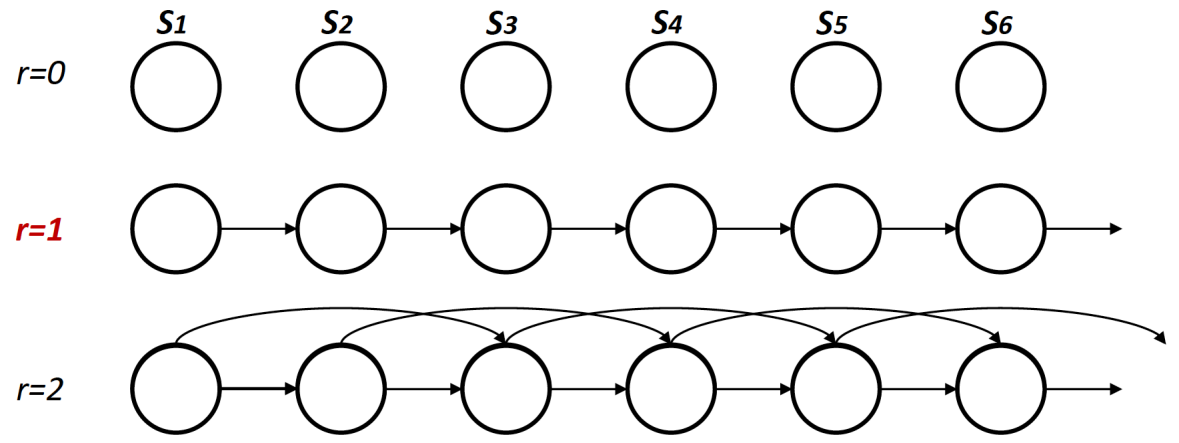
  — not on the previous states

1. Markov Model(MM): Markov Property

- Assumption: last k states are sufficient
- Process:

  — (r=1) first-order Markov process: $P(S_t|S_{t-1},\ldots,S_0) = P(S_t|S_{t-1})$

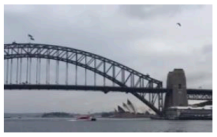
  — (r=2) second-order MP: $P(S_t|S_{t-1},\ldots,S_0) = P(S_t|S_{t-1},S_{t-2})$



- Example: P28-P31: ***REVIEW!!!!***

- Classify (Sydney) weather into three states
  - State 1: Rainy
  - State 2: Cloudy
  - State 3: Sunny


State1


State2


State3

- Assume that we examined the weather of Sydney for a year, and found following weather change pattern.

		Tomorrow		
		Rainy	Cloudy	Sunny
Today	Rainy	0.4	0.3	0.3
	Cloudy	0.2	0.6	0.2
	Sunny	0.1	0.1	0.8

$$S_{ij} = P(S_t = j \mid S_{t-1} = i)$$

$$S_{rainysunny} = 0.3$$

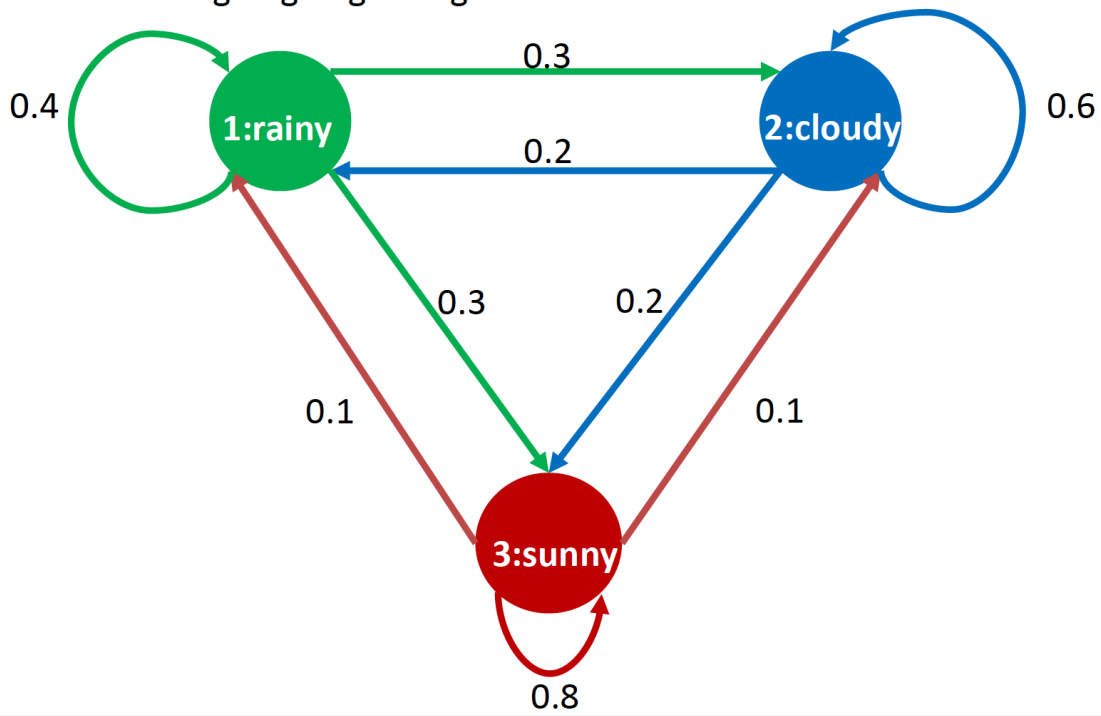
Assumption: Tomorrow weather depends only on today's!

Markov Model (MM): Example

		Tomorrow		
		Rainy	Cloudy	Sunny
Today	Rainy	0.4	0.3	0.3
	Cloudy	0.2	0.6	0.2
	Sunny	0.1	0.1	0.8

Visual illustration with diagram

- Each state corresponds to one observation
- Sum of outgoing edge weights is one



Markov Model (MM): Example

State Transition Matrix

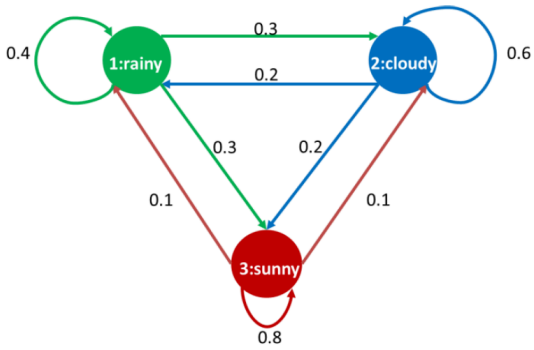
$$S_{ij} = P(S_t = j \mid S_{t-1} = i) \qquad 1 \le i,\ j \ge N$$

$$S_{ij} \ge 0$$

$$S = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1N} \\ S_{21} & S_{22} & \dots & S_{2N} \\ S_{31} & S_{32} & \dots & S_{3N} \\ \vdots & \vdots & \vdots & \vdots \\ S_{N1} & S_{N2} & \dots & S_{NN} \end{bmatrix}$$

		Tomorrow		
		Rainy	Cloudy	Sunny
Today	Rainy	0.4	0.3	0.3
	Cloudy	0.2	0.6	0.2
	Sunny	0.1	0.1	0.8

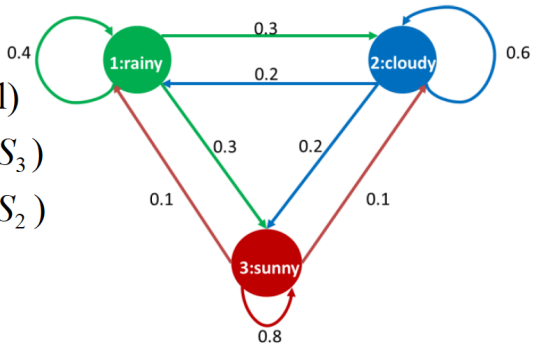
		Time *t+1*		
		S1	S2	S3
Time *t*	S1	0.4	0.3	0.3
	S2	0.2	0.6	0.2
	S3	0.1	0.1	0.8



Sequence Probability

- Question: What is the probability that the weather for the next 7 days will be "sun-sun-rain-rain-sun-cloudy-sun" when today is sunny?
  - S1: Rainy
  - S2: Cloudy
  - S3: Sunny

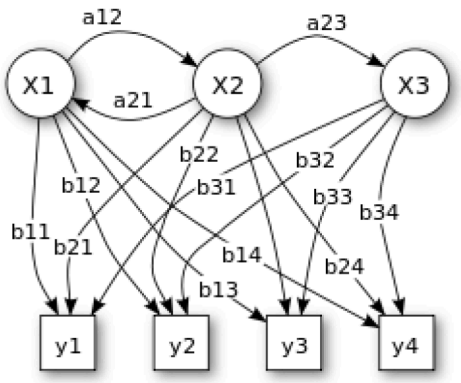
$$
\begin{aligned}
P(O \mid \text{model}) &= P(S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3 \mid \text{model}) \\
&= P(S_3) \cdot P(S_3 \mid S_3) \cdot P(S_3 \mid S_3) \cdot P(S_1 \mid S_3) \\
&\quad \cdot P(S_1 \mid S_1) P(S_3 \mid S_1) P(S_2 \mid S_3) P(S_3 \mid S_2) \\
&= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\
&= 1 \cdot (0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\
&= 1.536 \times 10^{-4}
\end{aligned}
$$



2. Hidden Markov Model (HMM)

— Concept :

- A class of probabilistic graphical model
- Predict a sequence of unknown (hidden) varibales based a set of observed variables



hidden

x	states ←
y	possible observations
a	state transition probabilities
b	output probabilities

- States are ***hidden***
- ***Observable events*** linked to states
- Each state has ***observation probabilities*** to determine the observable event

— Cal.: Joint Probs of a sequence of hidden states

- Initial state information (prior probability)

  — the initial probability of transitioning to a hidden state

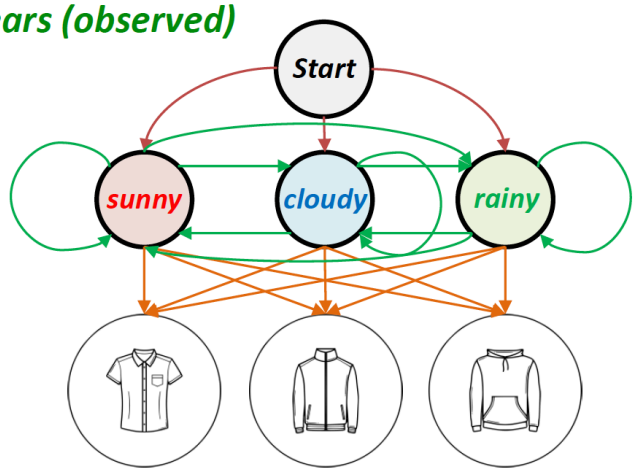
- Transition data

  — the probability of transitioning to a new state conditioned on a present state

- Emission data

  — the probability of transitioning to an observed state conditioned on a hidden state

Predicting the **weather *(hidden variable)*** based on the type of ***clothes that someone wears (observed)***



Priors

Rainy	0.6
Cloudy	0.3
Sunny	0.1

Transitions

		Tomorrow		
		Rainy	Cloudy	Sunny
Today	Rainy	0.6	0.3	0.1
	Cloudy	0.4	0.3	0.2
	Sunny	0.1	0.4	0.5

Emissions

	Rainy	Cloudy	Sunny
Shirts	0.8	0.19	0.01
Jacket	0.5	0.4	0.1
Hoodies	0.01	0.2	0.79

— Evaluation

What is the probability that the observations were generated by a given model?

— Observation sequence $X = x_1 x_2 x_3 \dots x_t$

— Model $\lambda = (A, B, \pi)$

— $\alpha_t(i)$: the t^{th} observation is on i state

— $b_i(x_t)$: the probability for x_t under the i state

- **Foward Algorithm**

  — Span a lattice of N states and T times

  — Keep the sum of probabilities of all the paths coming to each state i at time t.

  —Forward Probability

  note. S_j ***is a given state sequence***

  $P(O,Q|\lambda) = P(O|Q,\lambda) * P(Q|\lambda)$

  $P(O|Q,\lambda) = \Pi_{t=q}^{T} p(x_t|q_t,\lambda) = b_{q_1}(x_1)b_{q_2}(x_2)\dots b_{q_T}(x_T)$

  $P(Q|\lambda) = \pi_{q_1} b_{q_1}(x_1) a_{q_1 q_2} b_{q_2}(x_2) a_{q_2 q_3} b_{q_3}(x_3) \dots a_{q_{(T-1)} q_t} b_{q_T}(x_T)$

  $\alpha(j) = P(x_1 x_2 \dots x_t, q_t = S_j|\lambda)$

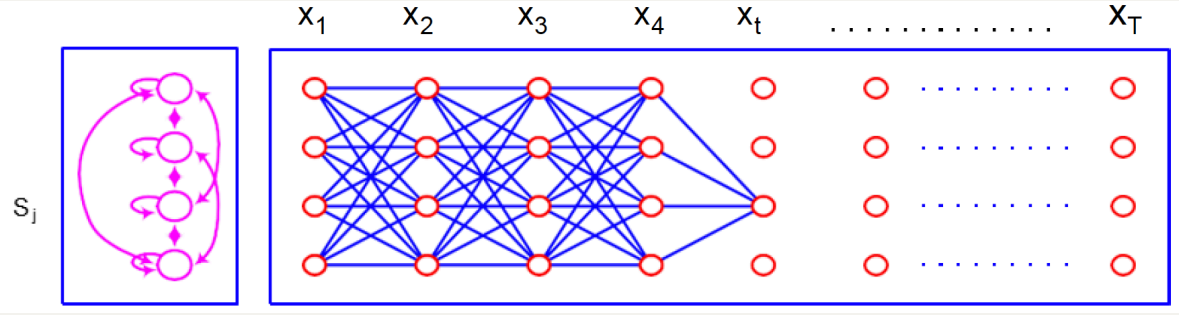
  $= \sum_{Q_t} P(x_1 x_2 \dots x_t, Q_t = q_1 \dots q_t|\lambda)$

  $= \sum_{i=1} \alpha_{t-1}(i) a_{ij} b_j(x_t)$

  - Initialiser: $\alpha_1(i) = \pi_i b_i(x_1) \dots\dots\dots 1 \le i \le N$
  - Induction

    $\alpha_t(j) = \sum_{i=1} \alpha_{t-1}(i) a_{ij} b_j(x_t) \dots 1 \le j \le N, t = 2, 3.., T$
  - Termination

    $P(X|\lambda) = \sum \alpha_T(i)$



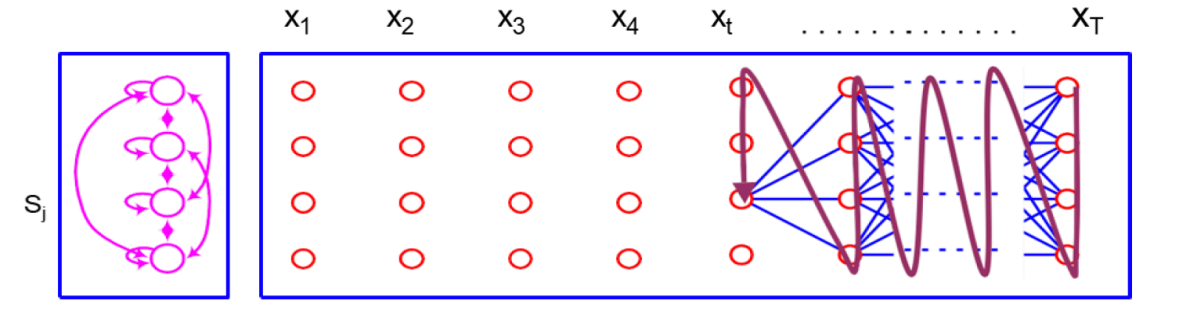
- **Backward Algorithm**

  — Initialization

  $\beta_T(i) = 1 \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots 1 \le i \le N$

  — Induction

$$\beta_t(i) = \sum a_{ij} b_j(x_{t+1}) \beta_{t+1}(j) \dots\dots\dots\dots\dots\dots 1 \le i \le N, t = T-1, T-2, \dots, 1$$



— Decoding

What is the most likely state observation, given a model and a sequence of observation?

— Given a model λ

— Observation sequence: $X = x_1, x_2, \dots, x_T$

— Calculation: $P(X, Q|\lambda) = ?$
$Q^* = arma x_Q P(X, Q|\lambda) = argma x_Q P(X|Q, \lambda) P(Q|\lambda)$

— A path or state sequence: $Q = q_1, L, q_T$

- **Viterbi Path**

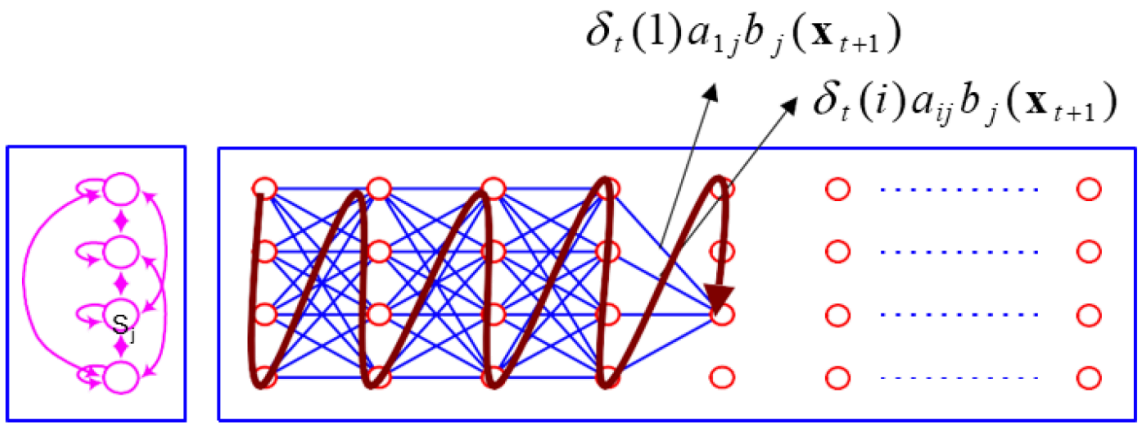
  — Span a lattice of N states and T times

  — Keep the probability and the previous node of the most probable path coming to each state i at time t

  — Recursive path selection

  a. Path probability
  b. Path node

- *Path probability* $\delta_{t+1}(j) = \max_{1 \le i \le N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1})$
- *Path node* $\psi_{t+1}(j) = \arg\max_{1 \le i \le N} \delta_t(i) a_{ij}$



- **Viterbi Algorithm**

Introduction

$$\delta_1(i) = \pi_i b_i(\mathbf{x}_1)$$
$$\psi_1(i) = 0$$

Recursion

$$\delta_{t+1}(j) = \max_{1 \le i \le N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1})$$
$$\psi_{t+1}(j) = \arg\max_{1 \le i \le N} \delta_t(i) a_{ij}$$

Termination

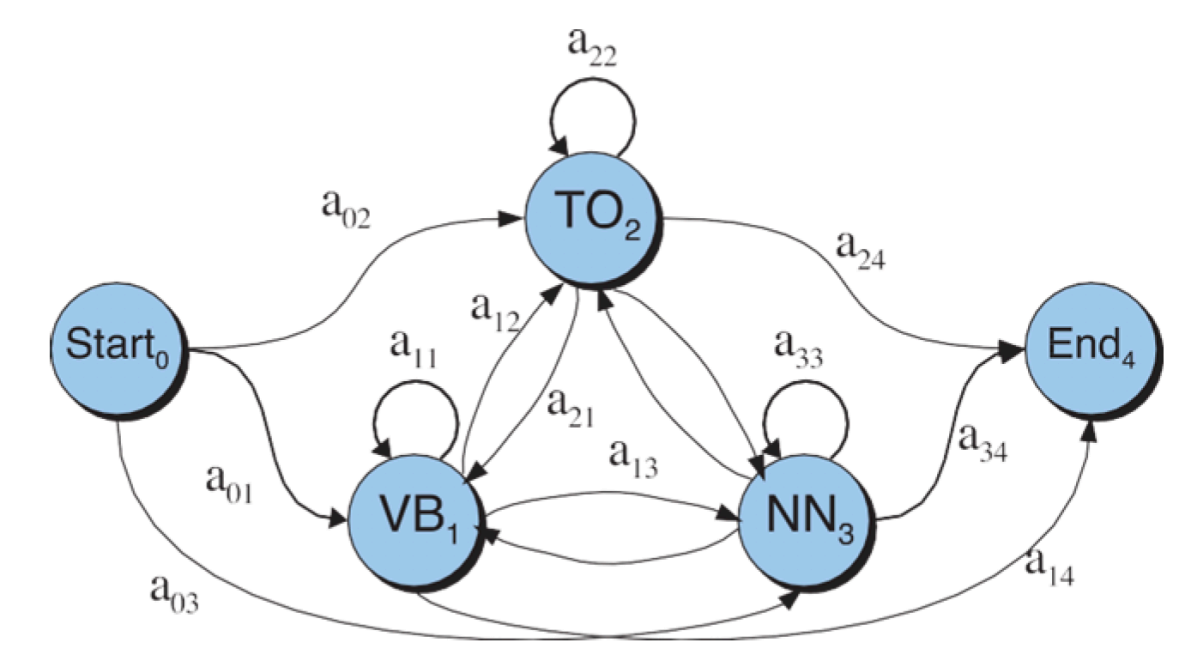
$$P^* = \max_{1 \le i \le N} \delta_T(i)$$
$$q_T^* = \arg\max_{1 \le i \le N} \delta_T(i)$$

Path Backtracking

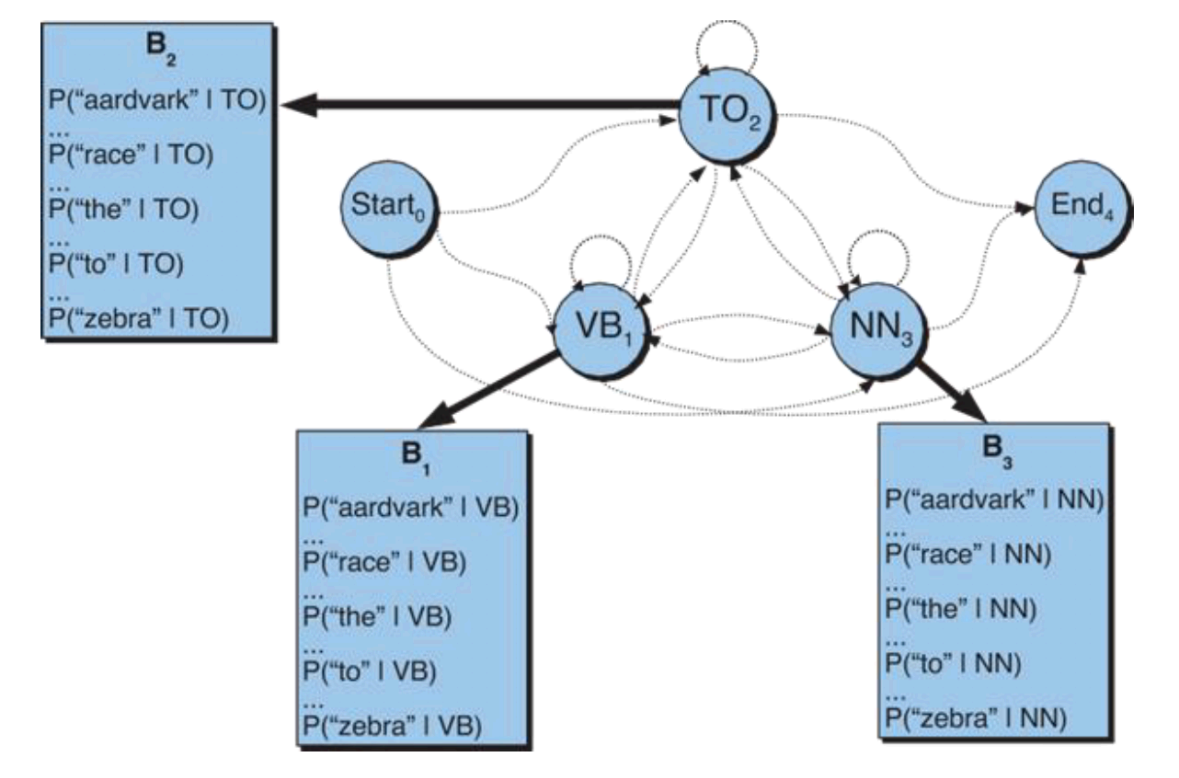
$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, \mathrm{K}, 1$$

4. HMM with POS Tagging

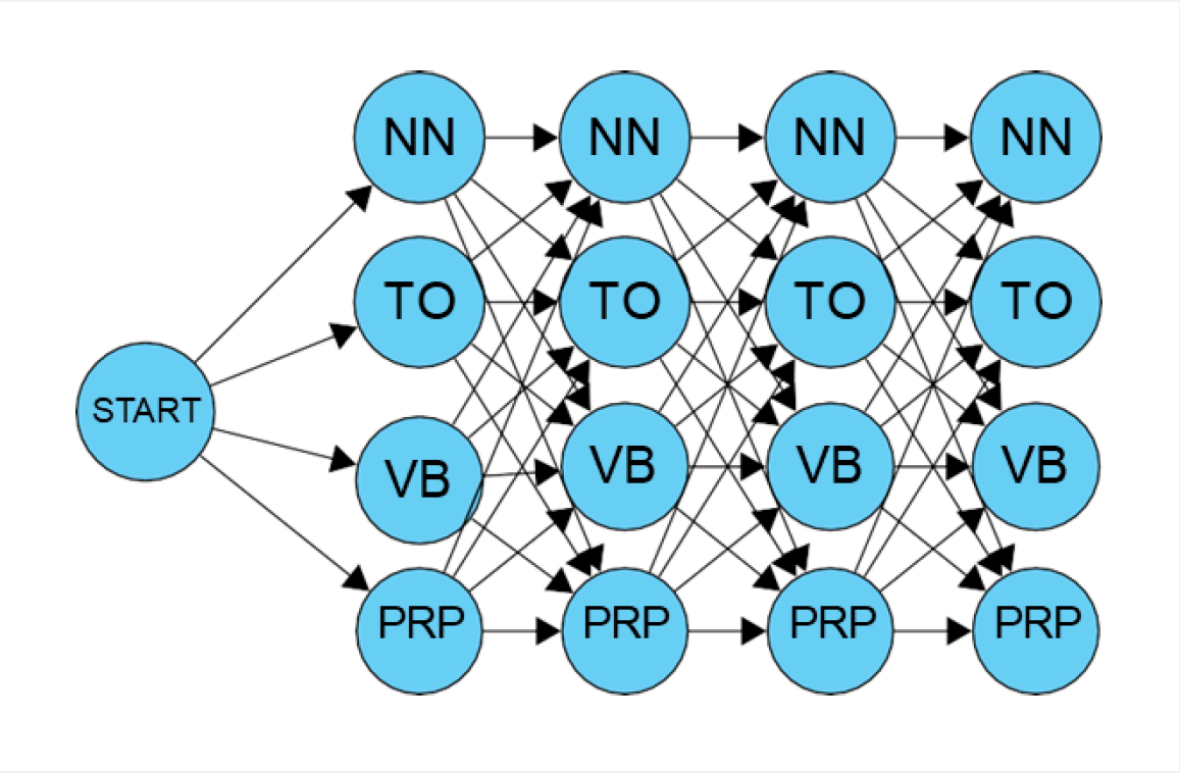
- Transition



- Emission



- Trellis



5. HMM Tagger Issue

- Unknown Words (OOV)

  — Solution 1: Use n-gram predict the correct tag

  — Solution 2: Use morphology (prefixes, suffixes) or hypenation

- Independency Problem

  — Only depend on every state and its corresponding observed object

  — Sequence labelling: having relation to individual words, also to the observed sequence length, word context and others

Maximum Entropy Markov Models(MEMM)

- Better expresion ability

  — Considering the dependencies between neighboring state and the entire observed sequence

- Reducing workload and keep consitency

  — Dosen't consider $P(X)$

— Reducing the modeling workload

— Learning the consistency between the target and the estimated function

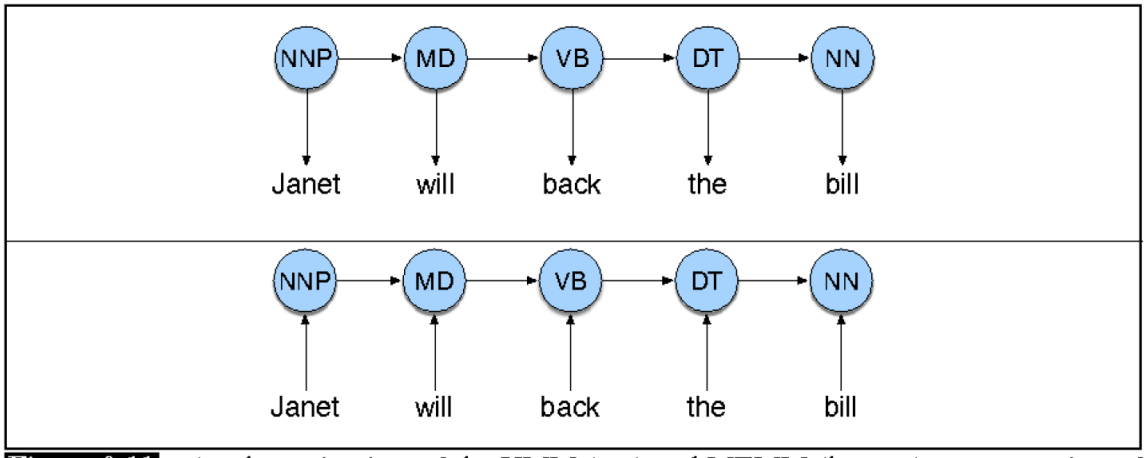
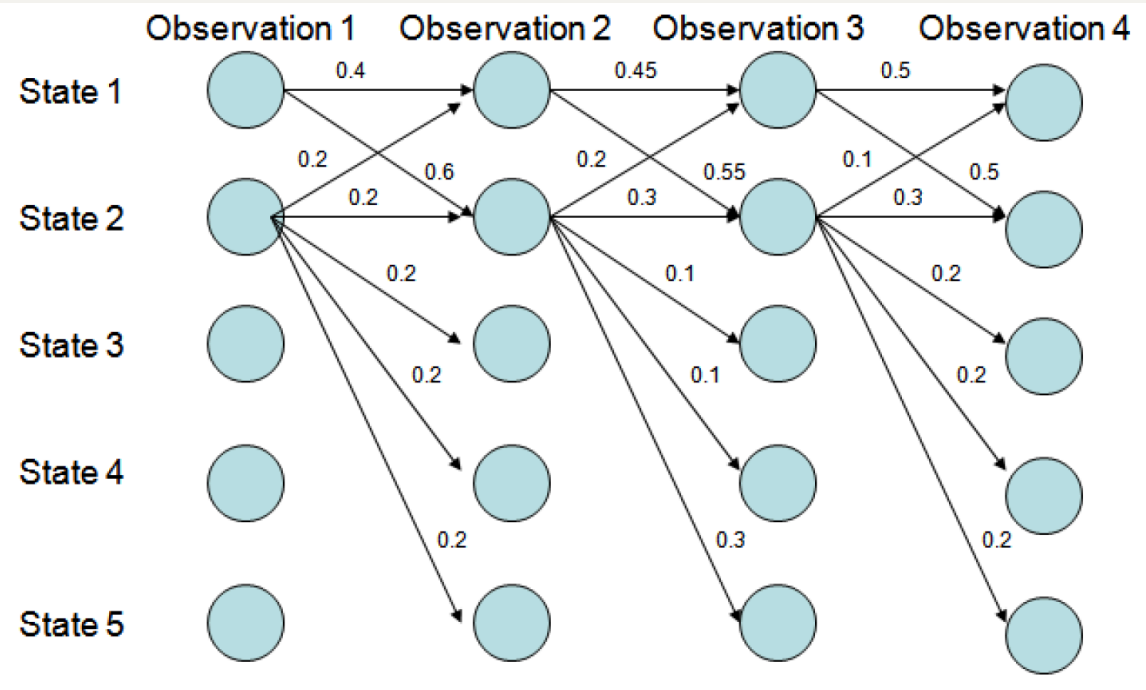


Figure 9.11 A schematic view of the HMM (top) and MEMM (bottom) representation of the probability computation for the correct sequence of tags for the *back* sentence. The HMM computes the likelihood of the observation given the hidden state, while the MEMM computes the posterior of each state, conditioned on the previous state and current observation.

- Label Bias Issue



Conditional Radonm Field

Attribute Comparison (HMM, MEMM, CRF)

- CRF addressed the labeling bias issue and eliminated unreasonable hypothesis in HMM
- MEMM adopts local variate normalization
- CRF adopts global variance normalisation

Model Comparison

- Generative Model

  — a model generate observed data randomly

  — model the joint probability p(x,y)

- Discriminative model (CRF)

  — directly estimate the posterior probability p(x|y)

  — aim at modeling the" discrimination" between different outputs

- Topological structure

  — HMM and MEMM are a directed graph

  — CRF is an undirected graph

Global optimum or local optimum comparison

- HMM

  — directly models the transition probability

  — directly models phenotype probability (emission)

  — cal. the probability of co-occurence

- MEMM

  — establish the probability of co-occurrence based on the transition probability and the phenotype probability

  — cal. the conditional probability and only adopts the local variance normalization, making it easy to fall into a local optimum

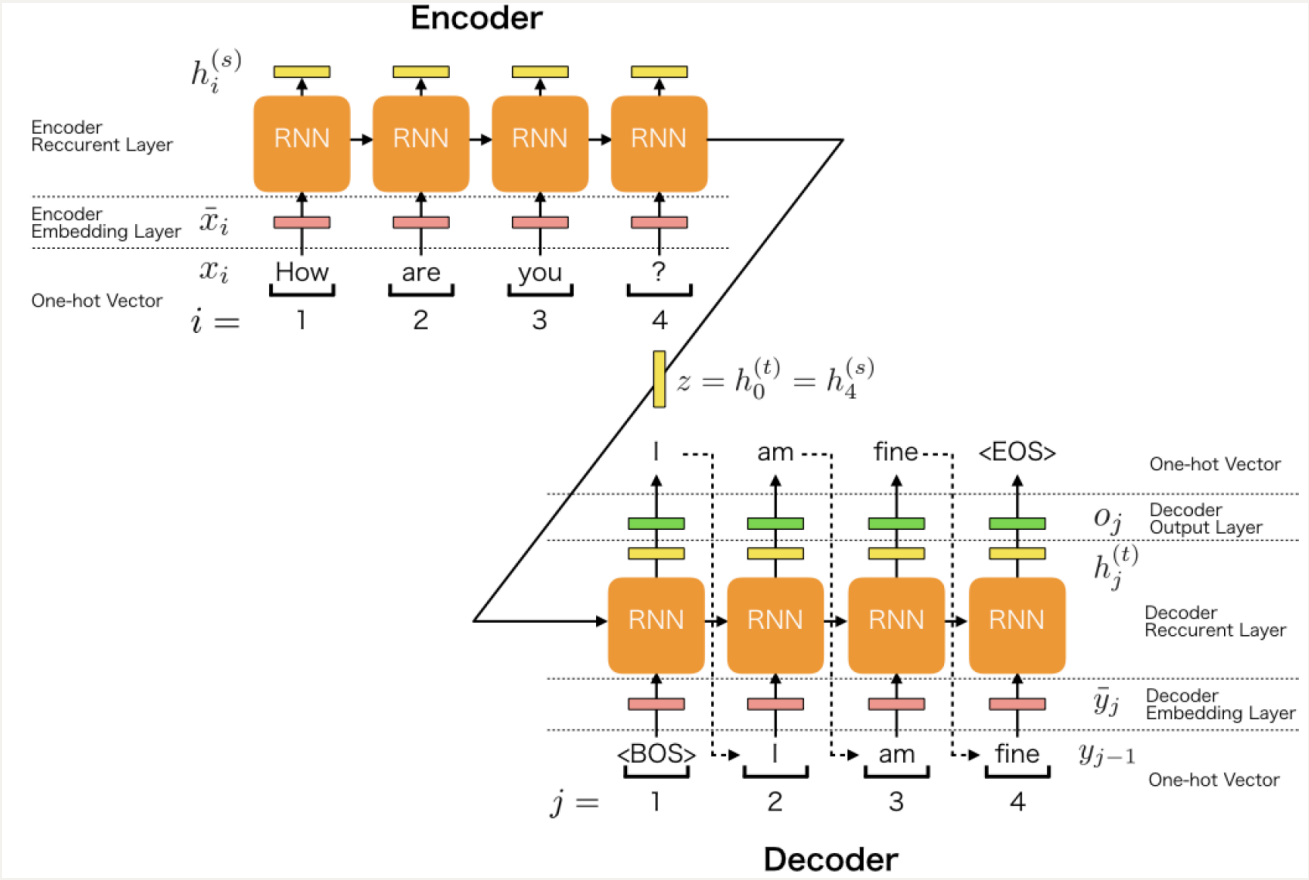
- CRF
  — cal. the normalization probability in the global scope
  — optimal global solution and resolves the labeling bias issue

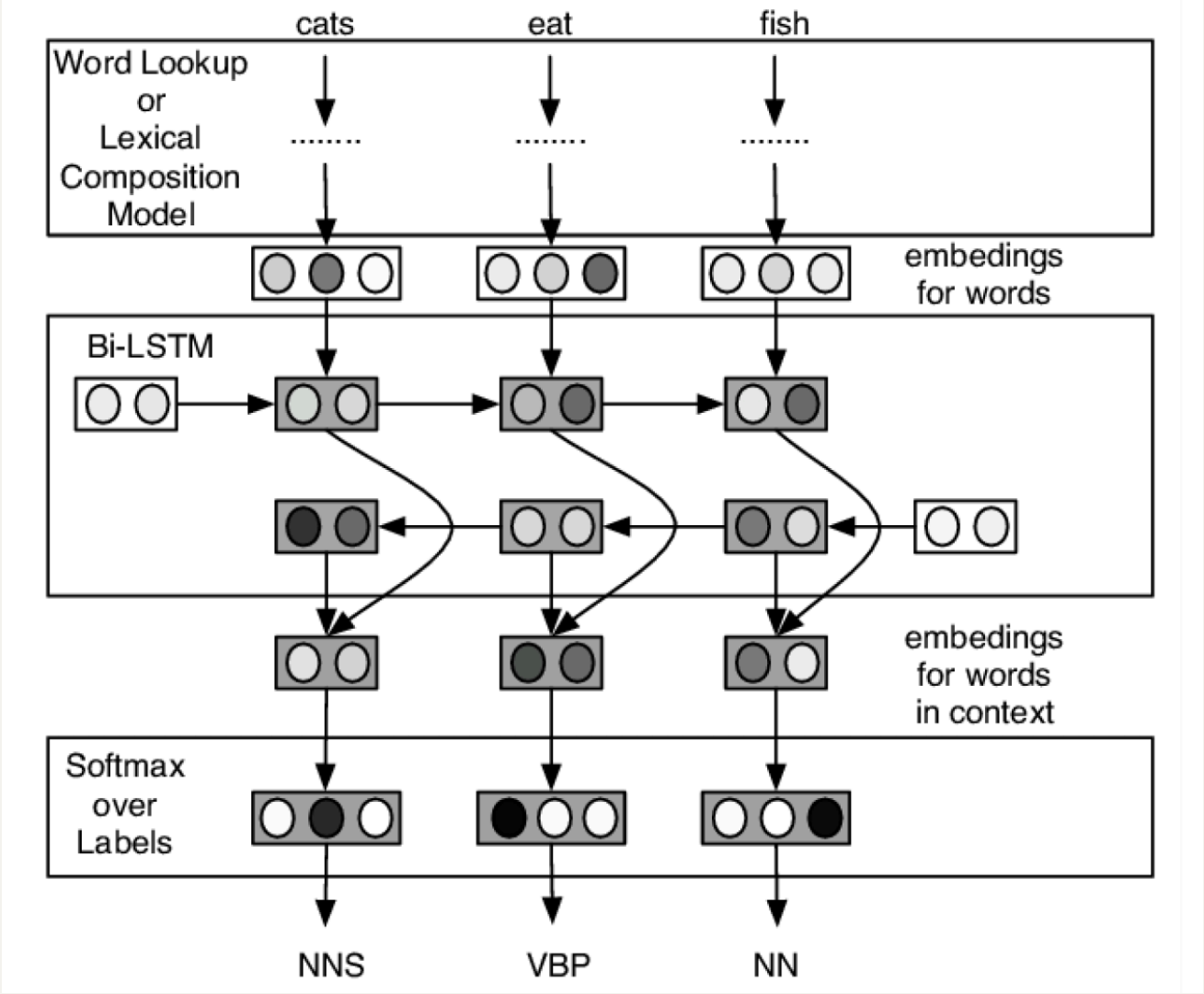
CRF Advantages and disadvantage

- with HMM
  — no strict independence assumptions
  — can accommodate any context information
- with MEMM
  — computs the conditinoal probabilities of global output nodes
  — overcoms the drawbacks of label bias in MEMM

- disadv.
  — computationally complex at the training stage of the algorithm
  — very difficult to re-train the model with updated new data

Deep learning approaches


Encoder
$h_i^{(s)}$
Encoder Reccurent Layer
RNN
RNN
RNN
RNN
Encoder Embedding Layer
$\bar{x}_i$
x_i
How
are
you
?
One-hot Vector
$i =$
1
2
3
4
$z = h_0^{(t)} = h_4^{(s)}$
I
am
fine
<EOS>
One-hot Vector
o_j
Decoder Output Layer
$h_j^{(t)}$
RNN
RNN
RNN
RNN
Decoder Reccurent Layer
$\bar{y}_j$
Decoder Embedding Layer
<BOS>
I
am
fine
y_{j-1}
One-hot Vector
$j =$
1
2
3
4
Decoder


cats
eat
fish
Word Lookup or Lexical Composition Model
embedings for words
Bi-LSTM
embedings for words in context
Softmax over Labels
NNS
VBP
NN

LSTM-based POS Tagging

Illustration of LSTM-based joint POS tagging and graph-based dependency parsing.