POS Tagging

Parts of speech

Parts of speech are constructed by grouping words that function similarly:

- with respect to the words that can occur nearby
- and by their morphological properties

The man _____ all the way home.

- Aristotle (384–322 BCE): the idea of having parts of speech a.k.a lexical categories, word classes, "tags", POS
- Dionysius Thrax of Alexandria (c. 100 BCE) : 8 parts of speech
  - noun, verb, article, adverb, preposition, conjunction, participle, pronoun

English parts of speech

- 8 parts of speech?
  - Noun (person, place or thing)
  - Verb (actions and processes)
  - Adjective (modify nouns)
  - Adverb (modify verbs)
  - Preposition (on, in, by, to, with)
  - Determiners (a, an, the, what, which, that)
  - Conjunctions (and, but, or)
  - Particle (off, up)

Brown corpus: 87 POS tags

Penn Treebank: ~45 POS tags

Open vs. Closed classes

- Open vs. Closed classes
  - Closed:
    - determiners: ***a, an, the***
    - pronouns: ***she, he, I***
    - prepositions: ***on, under, over, near, by, …***
    - Why "closed"?
  - Open:
    - Nouns, Verbs, Adjectives, Adverbs.

Closed vs. Open Class

- ***Closed class*** categories are composed of a small, fixed set of grammatical function words for a given language.
  - Pronouns, Prepositions, Modals, Determiners, Particles,
    Conjunctions
- Open class categories have large number of words and new ones are easily invented.
  - Nouns (Googler, textlish), Verbs (Google), Adjectives (geeky), Abverb (chompingly)

Open class (lexical) words

Nouns
Verbs

Proper
IBM
Italy
Mohan

Common
cat / cats
Snow
Kitaaba
,
kalama,

Main
see
registered
giraa,
gayaa

Adjectives *old older oldest*
sundara, acchaa,

Adverbs *slowly*
jaldii, teza

Numbers
122,312
one

... more

Closed class (functional)

Determiners *the some*

Conjunctions *and or*
aur, agar

Pronouns *he its*
Vaha, main

Modals
can
had

Prepositions *to with*

Particles *off up*
to, bhii, hii

Interjections *Ow Eh*

... more

Part Of Speech Tagging

- Annotate each word in a sentence with a part-of- speech marker.

John Saw the saw and decided to take it to the table.
NNP VBD DT NN CC VBD TO VB PRP IN DT NN

UDEP POS tags

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Ambiguity in POS Tagging

- "Like" can be a verb or a preposition
  - I like/VBP candy.
  - Time flies like/IN an arrow.
- "Around" can be a preposition, particle, or adverb
  - I bought it at the shop around/IN the corner.
  - I never got around/RP to getting a car.
  - A new Prius costs around/RB $25K.
- What is the POS for "back"?
  - The *back* door
  - On my *back*
  - Win the voters *back*
  - Promised to *back* the bill

POS Tagging Approaches

- **Rule-Based**: Human crafted rules based on lexical and other linguistic knowledge.
- **Learning-Based**: Trained on human annotated corpora like the Penn Treebank.
  - **Statistical models**: Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM), Conditional Random Field (CRF)
  - **Rule learning**: Transformation Based Learning (TBL)
  - **Neural networks**: Recurrent networks like Long Short Term Memory (LSTMs), Transformers !

Sequence Modelling
HMM

Probabilistic Sequence Models

- Probabilistic sequence models allow integrating uncertainty over multiple, interdependent classifications and collectively determine the most likely global assignment.

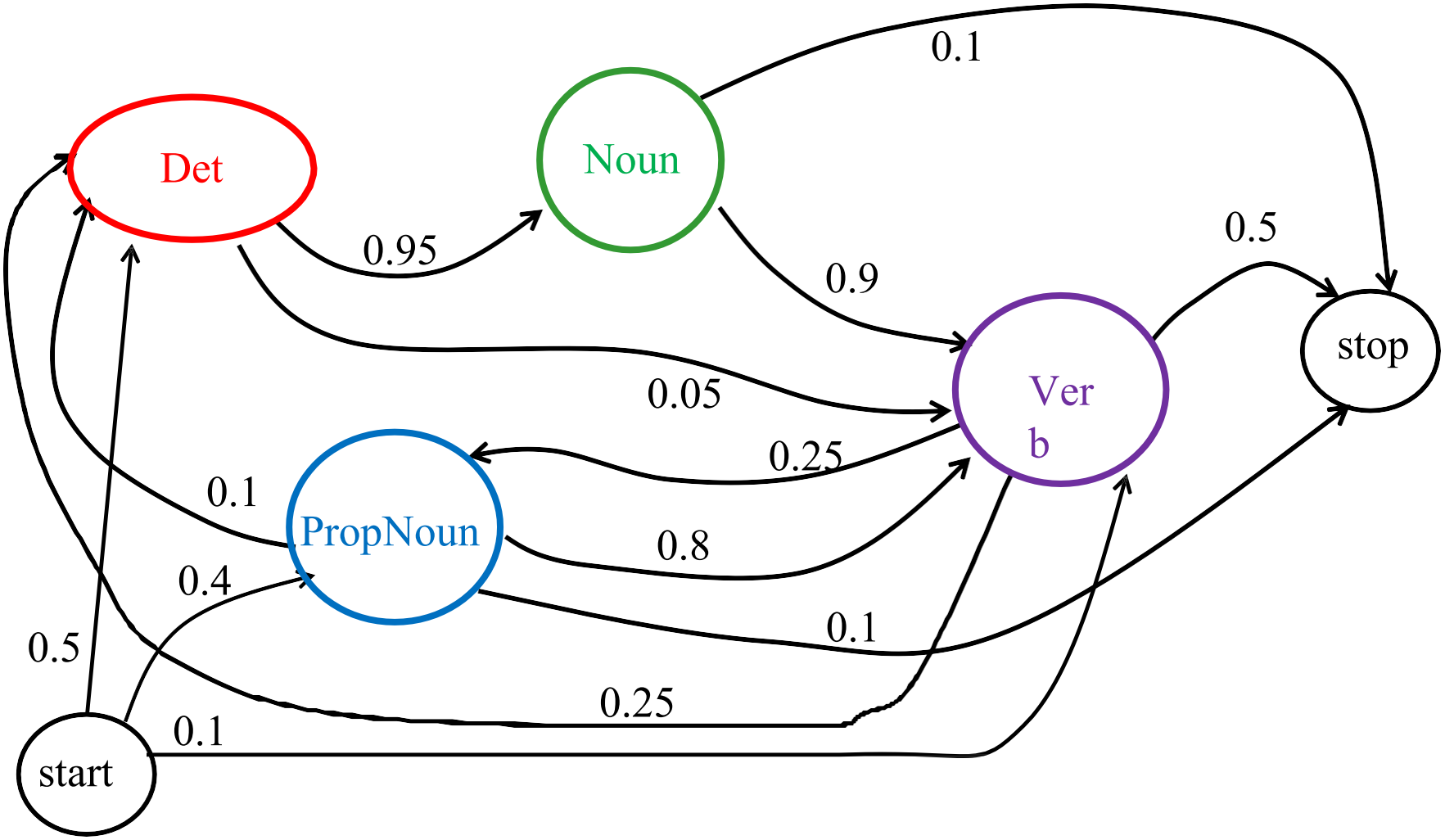
- Two standard models
  - Hidden Markov Model (HMM)
  - Conditional Random Field (CRF)

Markov Model / Markov Chain

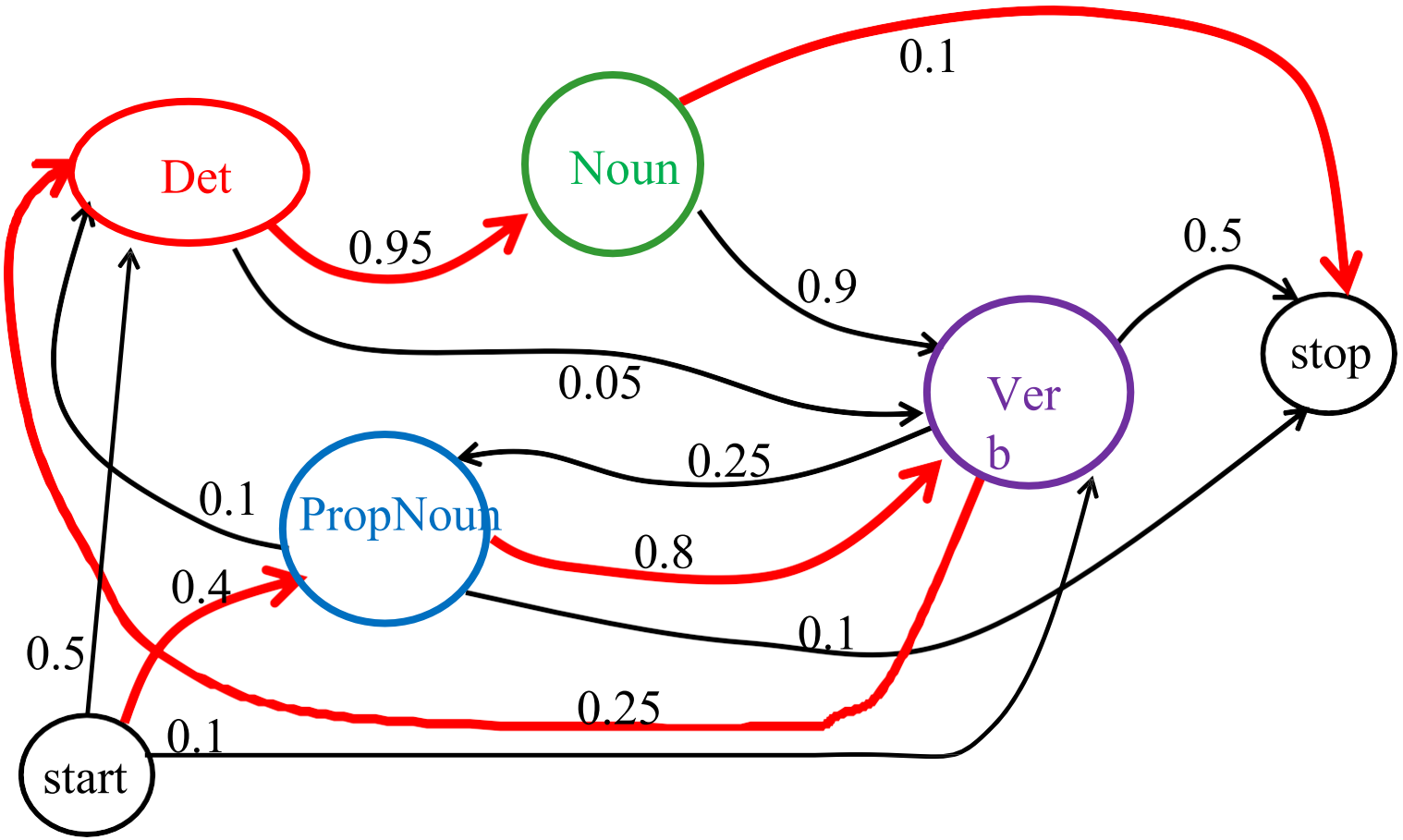
- A finite state machine with probabilistic state transitions.
- Markov assumption: the next state only depends on the current state and independent of previous history.

  - The set of all states: $\{s\}$
  - Initial states: S_I
  - Final states: S_F
  - Probability of making the transition from state i to j: a_{ij}

Sample Markov Model for POS



Sample Markov Model for POS



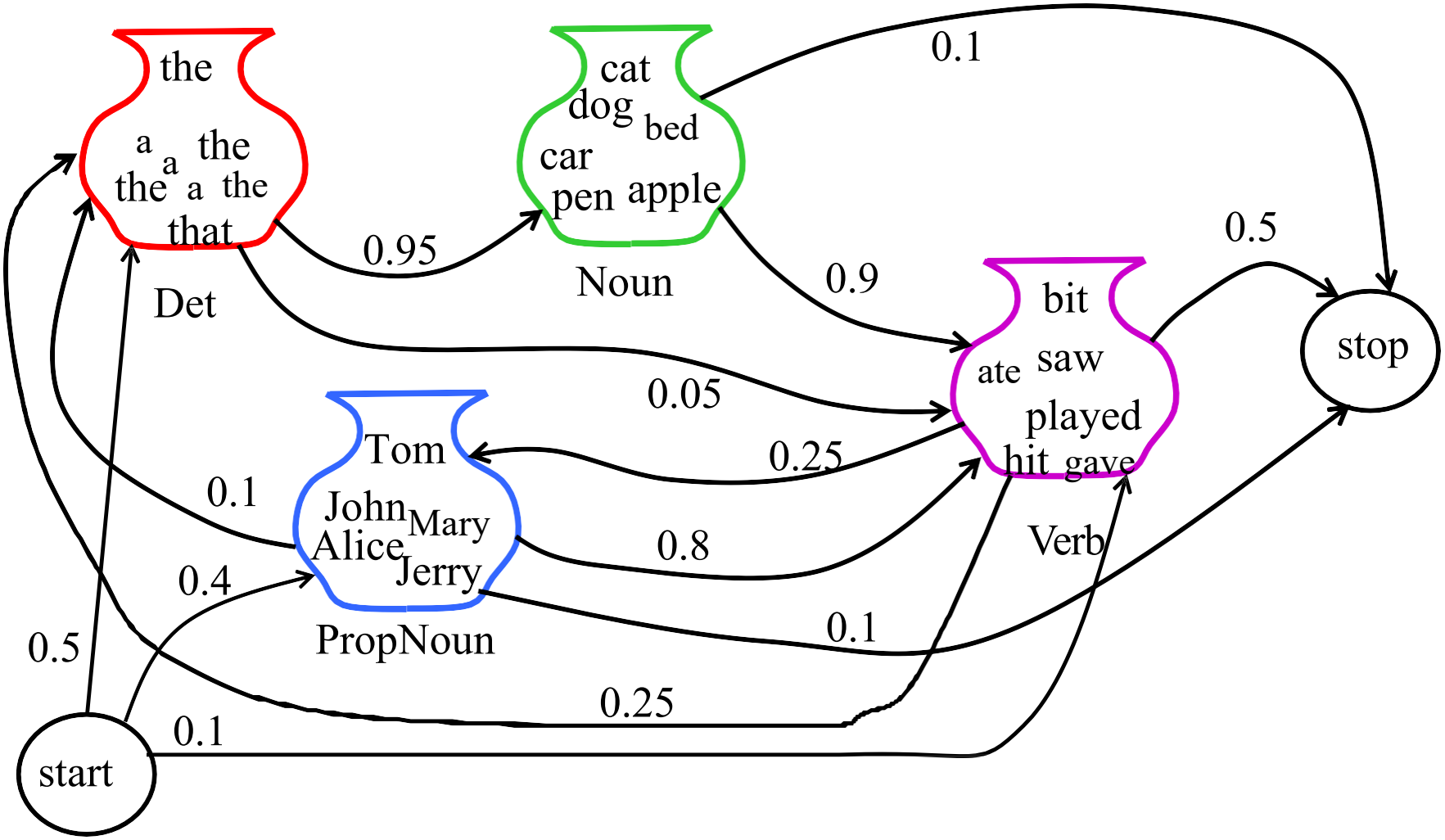
P(PropNoun Verb Det Noun) =
0.4∗0.8∗0.25∗0.95∗0.1=0.0076

Hidden Markov Model

- Probabilistic generative model for sequences.
- Assume
  - An underlying set of ***hidden*** states in which the model can be (e.g. parts of speech).
  - Probabilistic transitions between states over time
  - A ***probabilistic*** generation of tokens from states (e.g. words generated for each POS).

  - A set of output symbols
  - Probability of emitting the symbol k in state j: $b_j(k)$

Sample HMM for POS



Definition of an HMM

- A set of N +2 states $S=\{s_0,s_1,s_2, \ldots s_N, s_F\}$
- A set of M possible observations $V=\{v_1,v_2 \ldots v_M\}$
- A state transition probability distribution $A=\{a_{ij}\}$

$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i) \qquad 1 \le i, j \le N \text{ and } i = 0, j = F$$

$$\sum_{j=1}^{N} a_{ij} + a_{iF} = 1 \quad 0 \le i \le N$$

- Observation probability distribution for each state j $B=\{b_j(k)\}$

$$b_j(k) = P(v_k \text{ at } t \mid q_t = s_j) \quad 1 \le j \le N \quad 1 \le \mathrm{k} \le \mathrm{M}$$

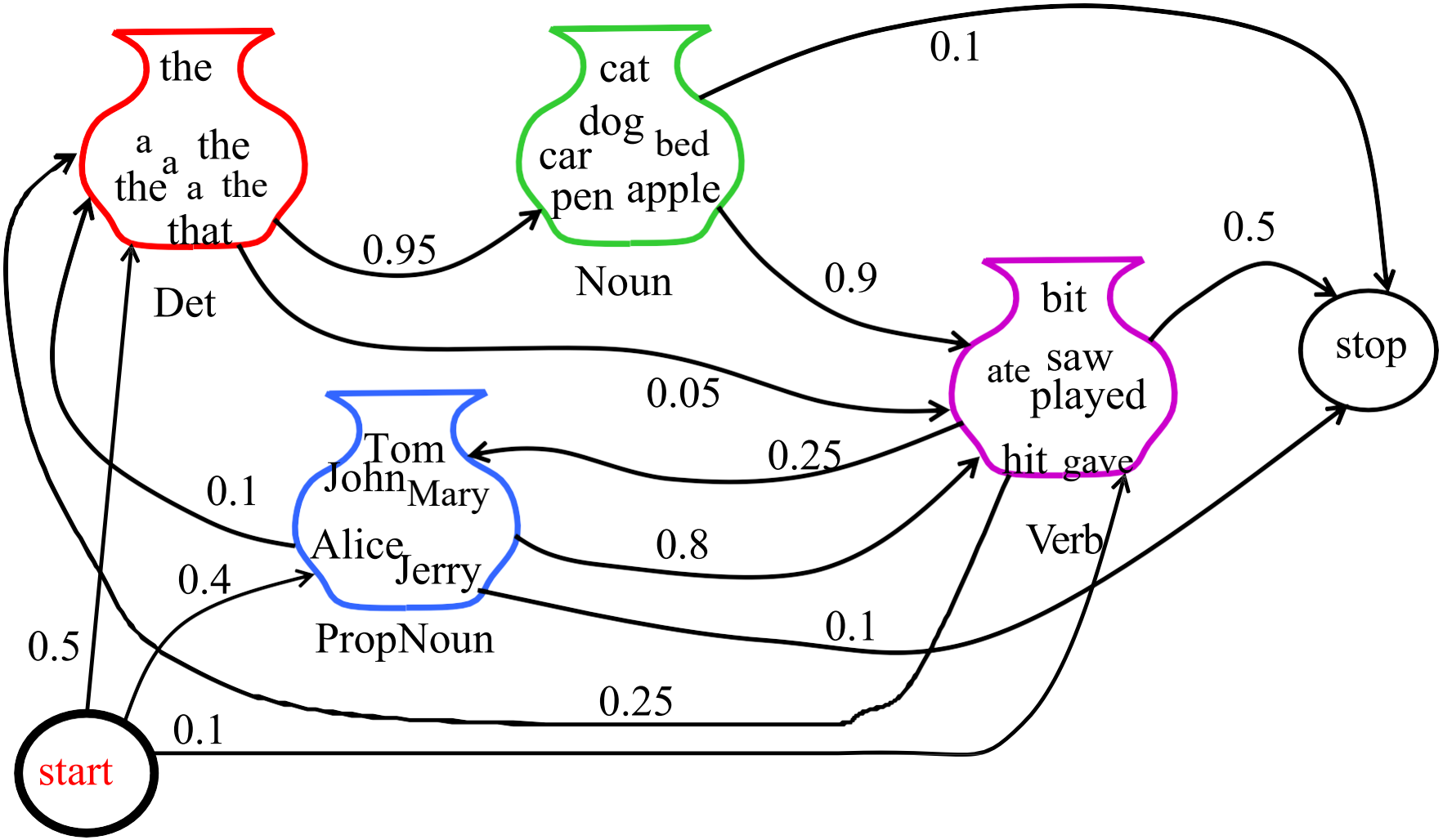
- Total parameter set
  $\lambda=\{A,B\}$

HMM Generation Procedure

- To generate a sequence of T observations:
  $O = o_1\ o_2\ \ldots\ o_T$

Set initial state $q_1 = s_0$
For $t = 1$ to T
    Transit to another state $q_{t+1} = s_j$ based on transition distribution a_{ij} for state q_t
    Pick an observation $o_t = v_k$ based on being in state q_t using distribution $b_{qt}(k)$

Sample HMM Generation



Three Useful HMM Tasks

1. Observation Likelihood (Evaluation): given a model and an output sequence, what is the probability that the model generated that output?
2. Most likely state sequence (Decoding): given a model and an output sequence, what is the most likely state sequence through the model that generated the output?
3. Maximum likelihood training (Learning): given a model and a set of observed sequences, how do we set the model's parameters so that it has a high probability of generating those sequences?

HMM: Observation Likelihood

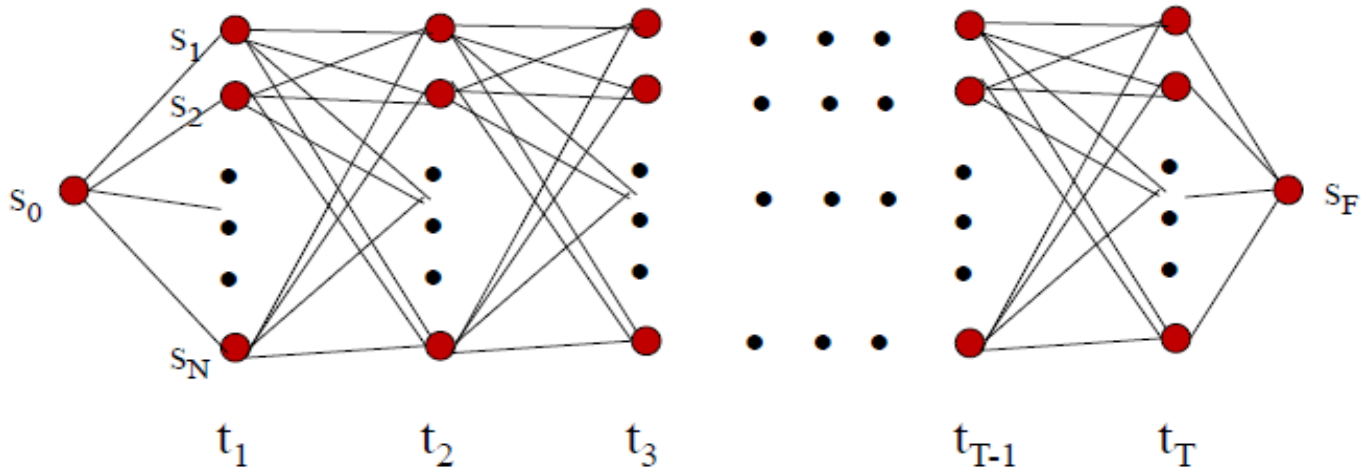
- Given a sequence of observations, O, and a model with parameters, λ, what is the probability that this observation was generated by this model:
- $P(O|\lambda)$?

- Useful for two tasks:
  - Sequence Classification
  - Most Likely Sequence

HMM: Observation Likelihood Efficient Solution

- Forward Algorithm: Compute a ***forward trellis*** that compactly and implicitly encodes information about all possible state paths.

Forward Trellis



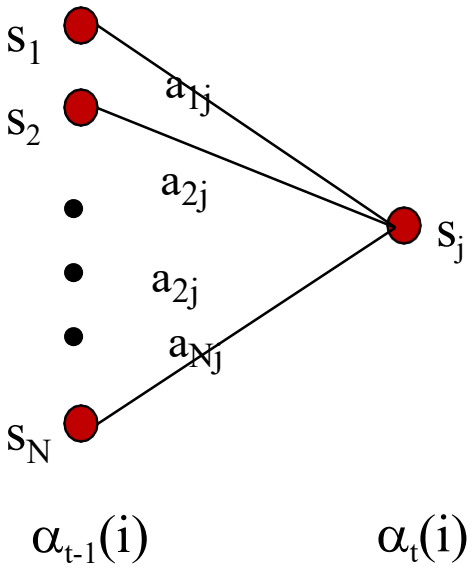
Continue forward in time until reaching final time point and sum probability of ending in final state.

Forward Probabilities

- Let $\alpha_t(j)$ be the probability of being in state j after seeing the first t observations (by summing over all initial paths leading to j).

$$\alpha_t(j) = P(o_1, o_2, \ldots o_t, q_t = s_j \mid \lambda)$$

Forward Step



- Consider all possible ways of getting to s_j at time t by coming from all possible states s_i and determine probability of each.
- Sum these to get the total probability of being in state s_j at time t while accounting for the first $t-1$ observations.
- Then multiply by the probability of actually observing o_t in s_j.

Computing Forward Probabilities

- Initialization

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$

- Recursion

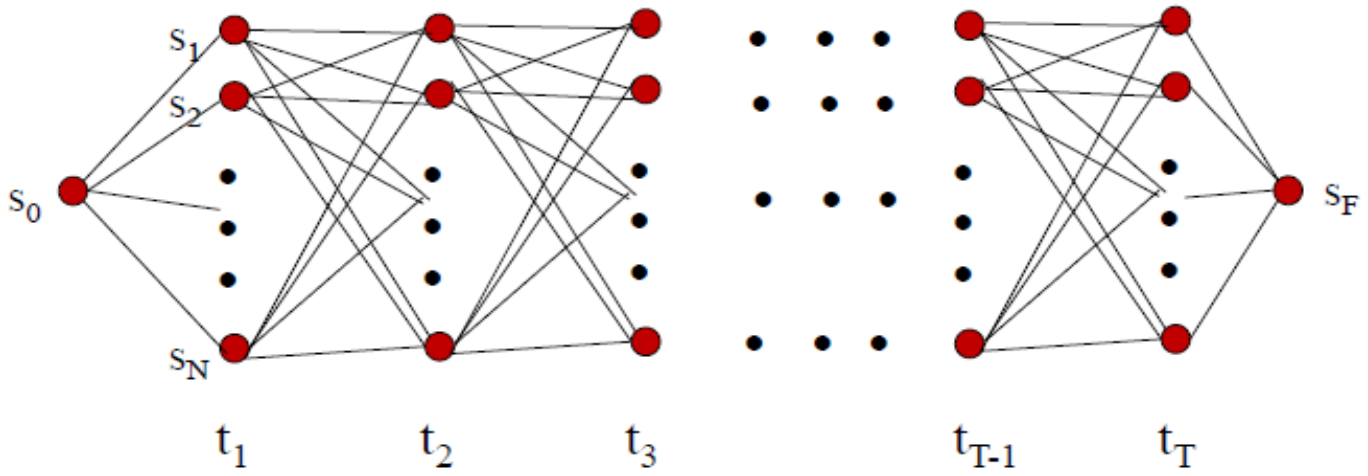
$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 1 \le j \le N, \quad 1 < t \le T$$

- Termination

$$P(O \mid \lambda) = \alpha_{T+1}(s_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

$$\alpha_t(j) = P(o_1, o_2, \ldots o_t, q_t = s_j \mid \lambda)$$

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$



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

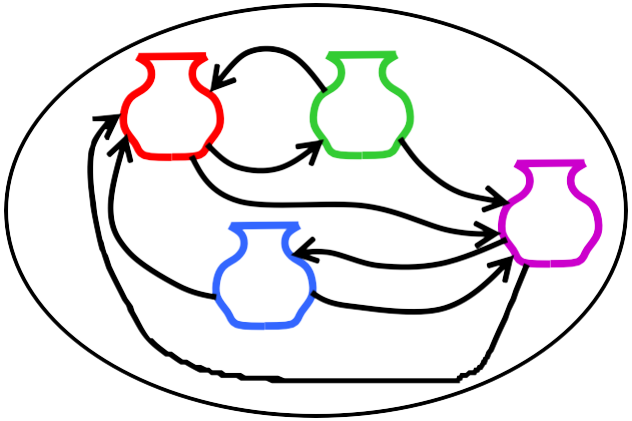
$$P(O \mid \lambda) = \alpha_{T+1}(s_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

Forward Computational Complexity

- Requires only $O(TN^2)$ time to compute the probability of an observed sequence given a model.

2. Most Likely State Sequence (Decoding)

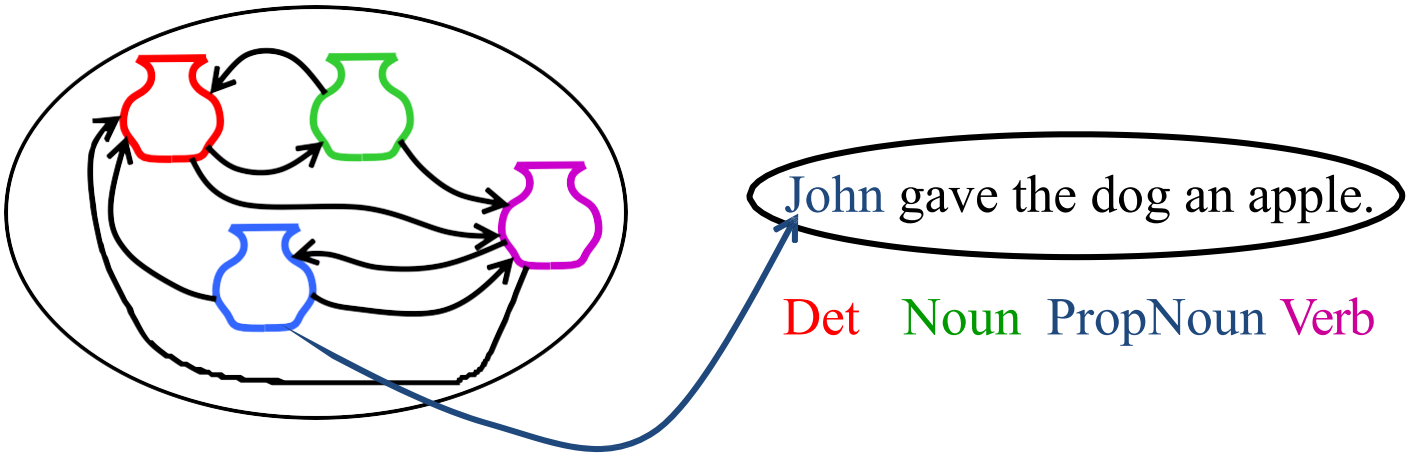
- Given an observation sequence, O, and a model, λ, what is *the most likely state sequence*, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



John gave the dog an apple.

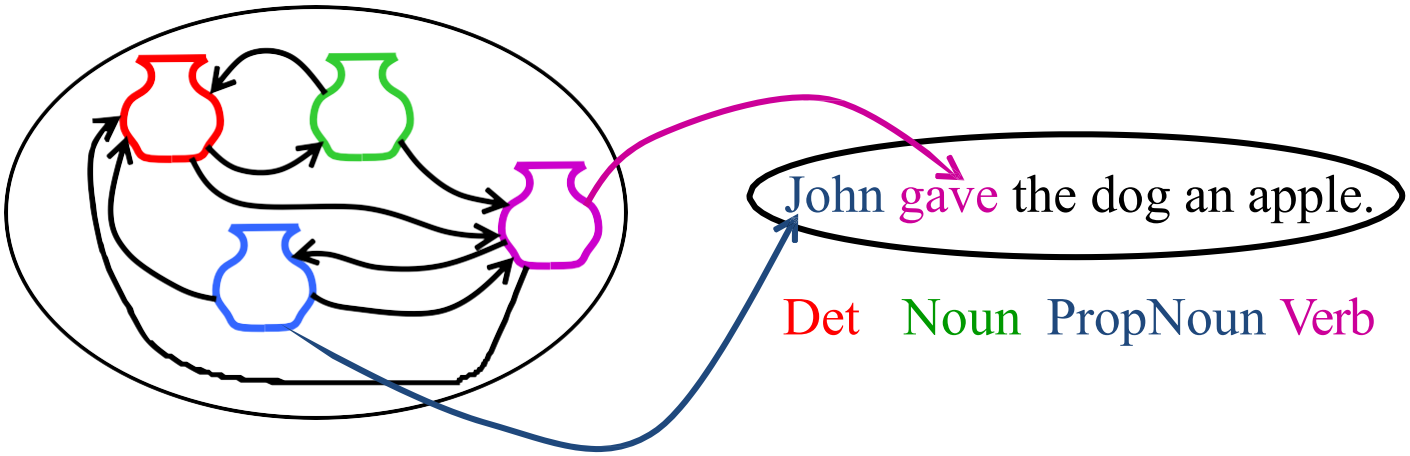
Most Likely State Sequence

- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



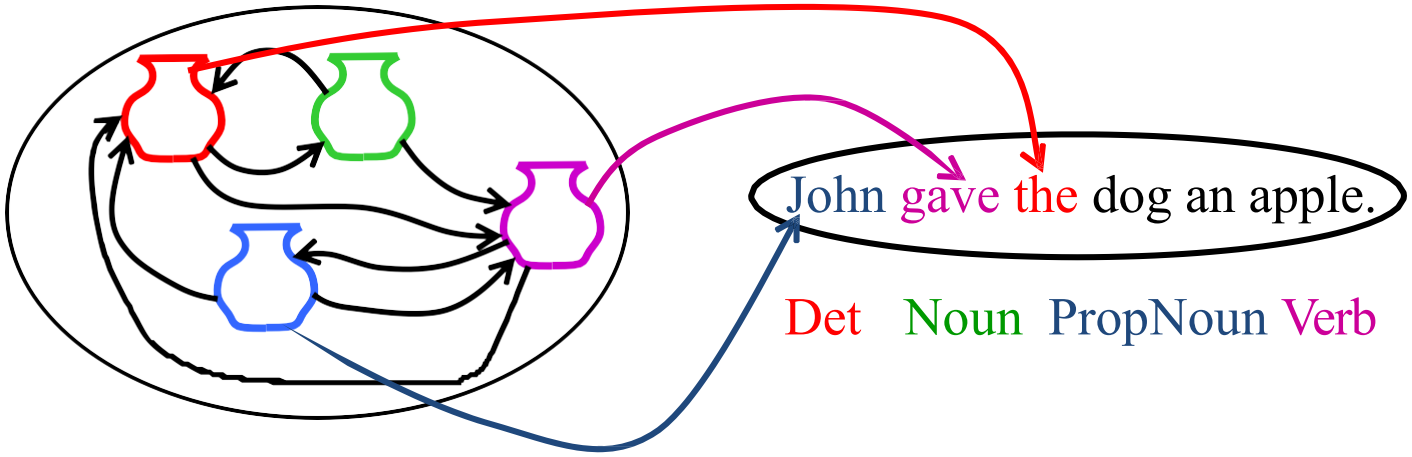
Most Likely State Sequence

- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



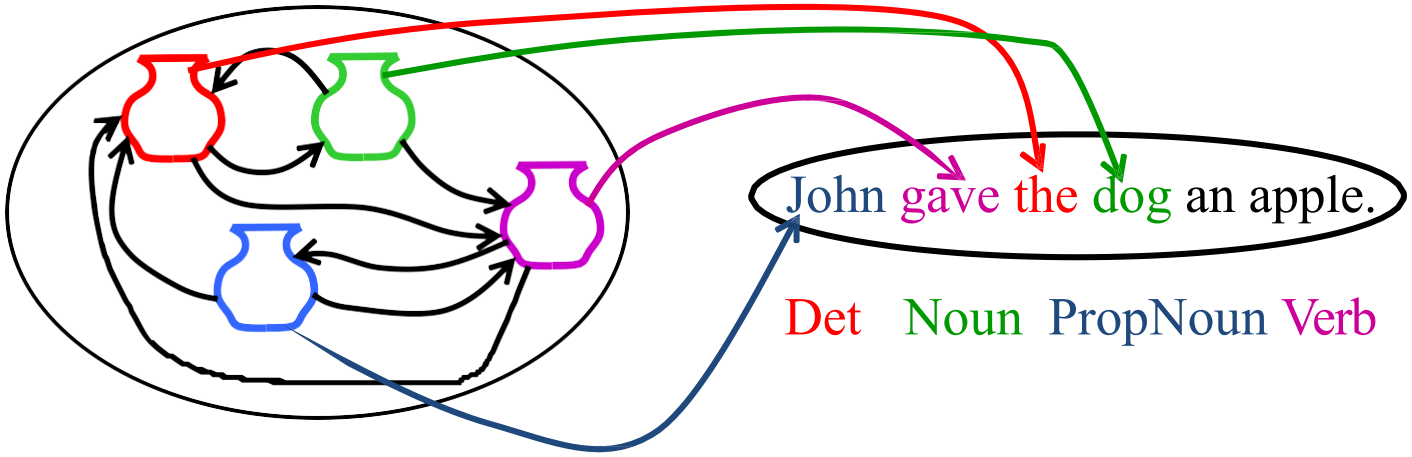
Most Likely State Sequence

- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



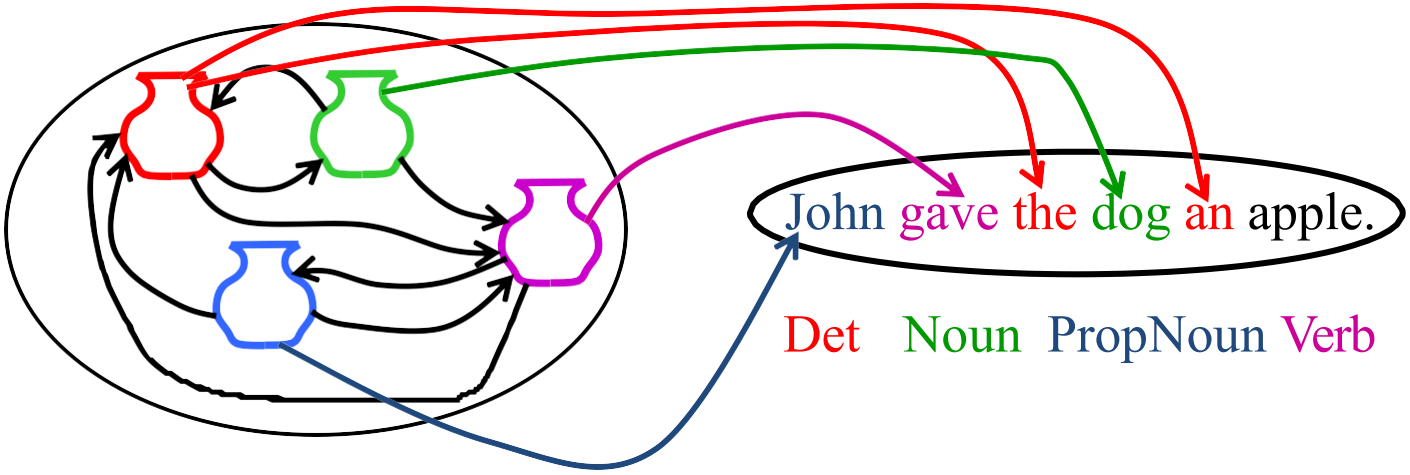
Most Likely State Sequence

- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



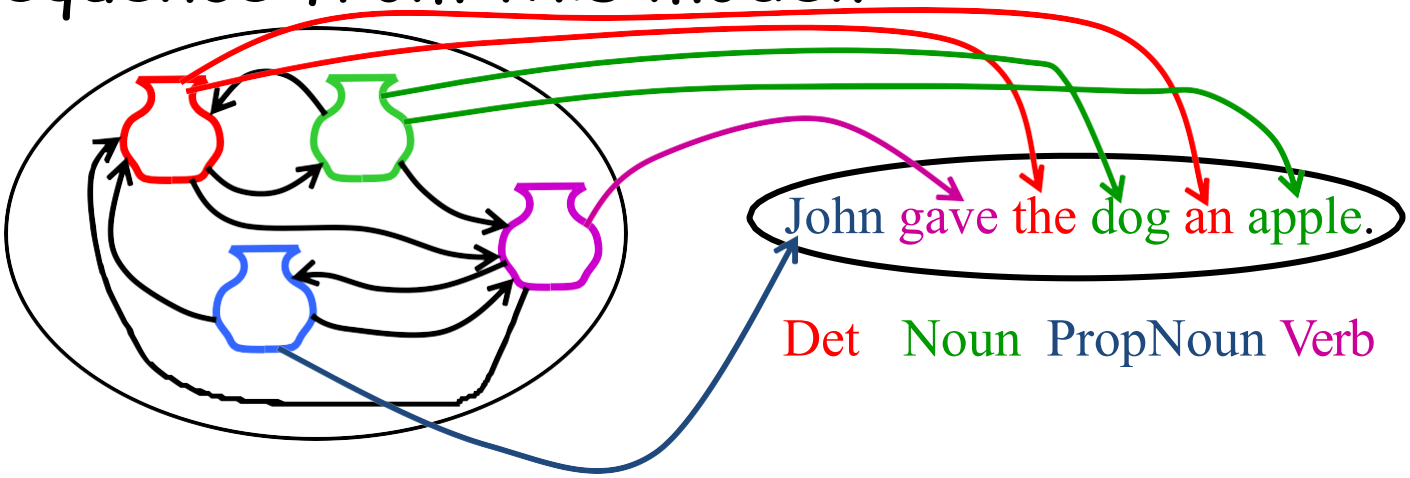
Most Likely State Sequence

- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



Most Likely State Sequence

- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



Viterbi Scores

- Recursively compute the probability of the most likely subsequence of states that accounts for the first t observations and ends in state s_j.

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1, \ldots, q_{t-1}, o_1, \ldots, o_t, q_t = s_j \mid \lambda)$$

  - Also record "backpointers" that subsequently allow backtracing the most probable state sequence.
    - $bt_t(j)$ stores the state at time $t-1$ that maximizes the probability that system was in state s_j at time t (given the observed sequence).

Computing the Viterbi Scores

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1, \ldots, q_{t-1}, o_1, \ldots, o_t, q_t = s_j \mid \lambda)$$

- Initialization $v_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$
- Recursion

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \quad 1 < t \le T$$

- Termination

$$P^* = v_{T+1}(s_F) = \max_{i=1}^{N} v_T(i) a_{iF}$$

Analogous to Forward algorithm except take *max* instead of sum

Computing the Viterbi Backpointers

- Initialization

$$bt_1(j) = s_0 \quad 1 \le j \le N$$

- Recursion

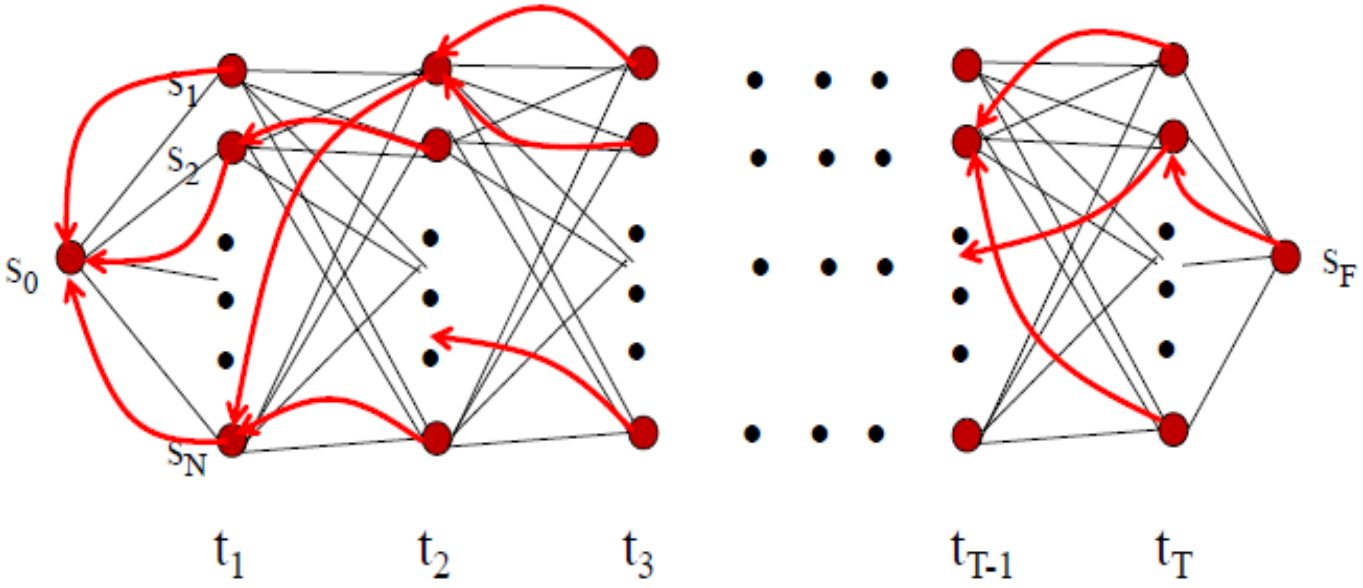
$$bt_t(j) = \underset{i=1}{\overset{N}{\operatorname{argmax}}} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, 1 \le t \le T$$

- Termination

$$q_T^* = bt_{T+1}(s_F) = \underset{i=1}{\overset{N}{\operatorname{argmax}}} v_T(i) a_{iF}$$

Final state in the most probable state sequence.
Follow backpointers to initial state to
construct full sequence.

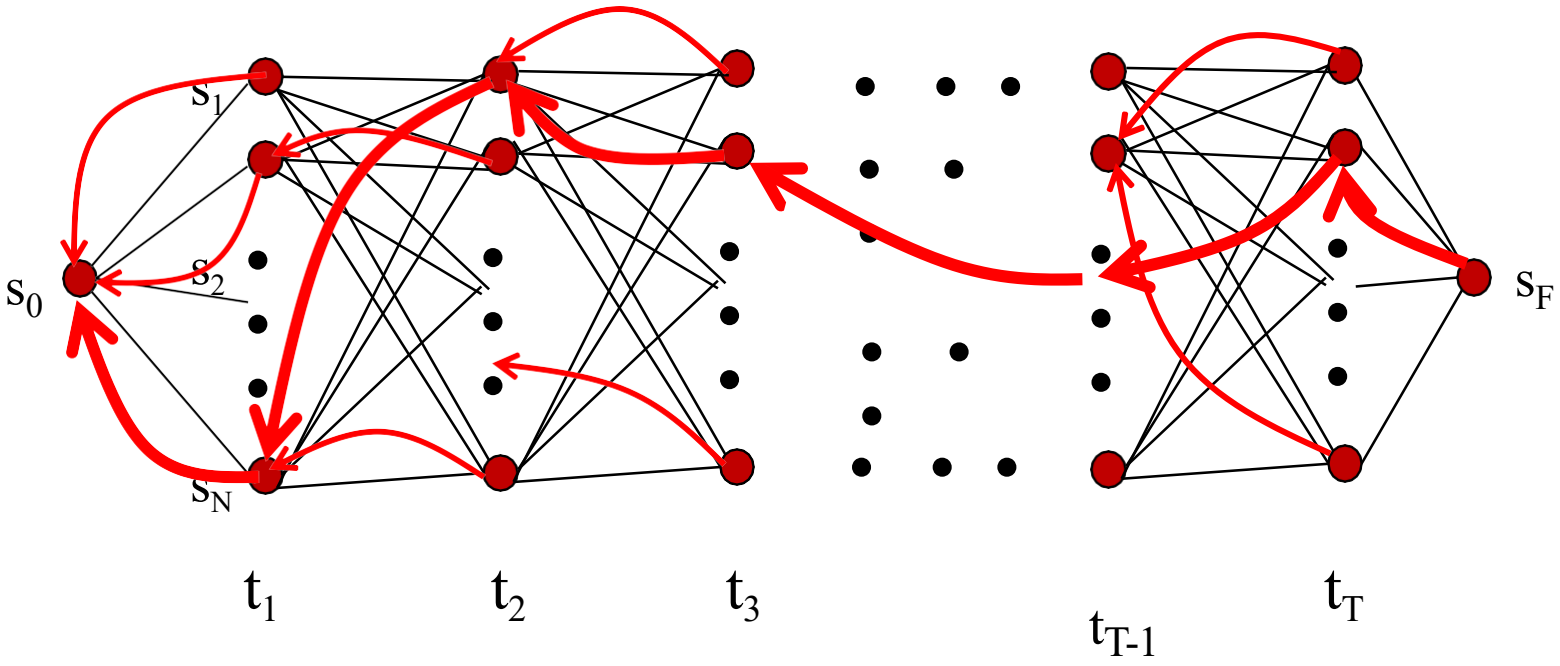
Viterbi Backpointers



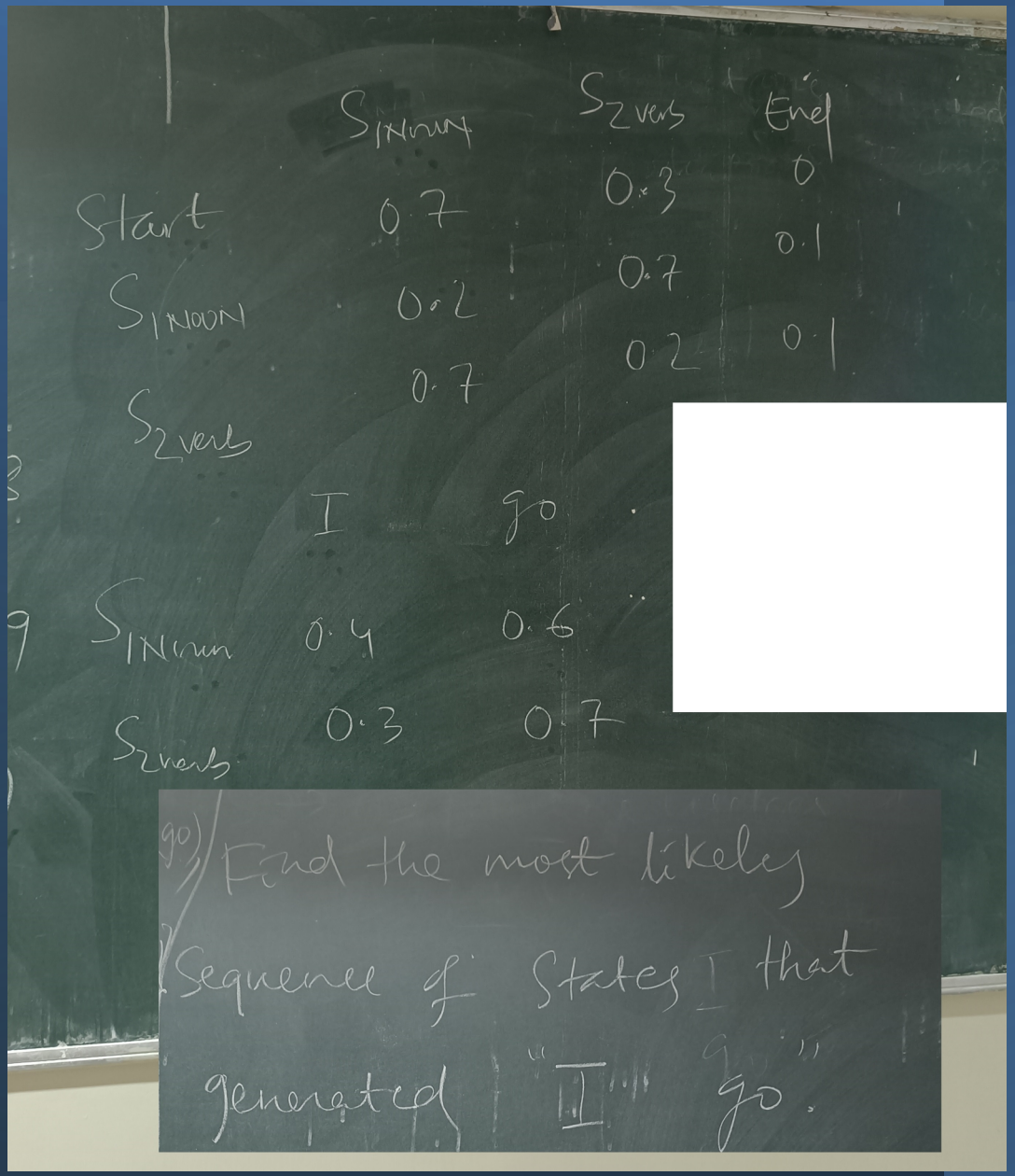
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \quad 1 < t \le T$$

$$bt_t(j) = \operatorname{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, 1 \le t \le T$$

Viterbi Backtrace



Most likely Sequence: $s_0\ s_N\ s_1\ s_2\ \ldots s_2\ s_F$


S1NOUN S2VERB END
Start 0.7 0.3 0
S1NOUN 0.2 0.7 0.1
S2VERB 0.7 0.2 0.1
I go
S1NOUN 0.4 0.6
S2VERB 0.3 0.7
go) Find the most likely sequence of states I that generated "I" "go".

$V_1(\text{noun}) = 0.7 \times 0.4 = .28$

$V_1(\text{verb}) = 0.3 \times 0.3 = 0.09$

$$V_2(\text{NOUN}) = \max\{ V_1(\text{NOUN}) * a_{NN} * b_N(\text{go}),$$

$$V_1(\text{verb}) * a_{VN} * b_N(\text{go}) \}$$

$$= \{ 0.28 * 0.2 * 0.6,$$

$$0.09 * 0.7 * 0.6 \}$$

$$= 0.0378$$

$$V_2(\text{verb}) = \max\{ V_1(N) * a_{NV} * b_V(\text{go}),$$

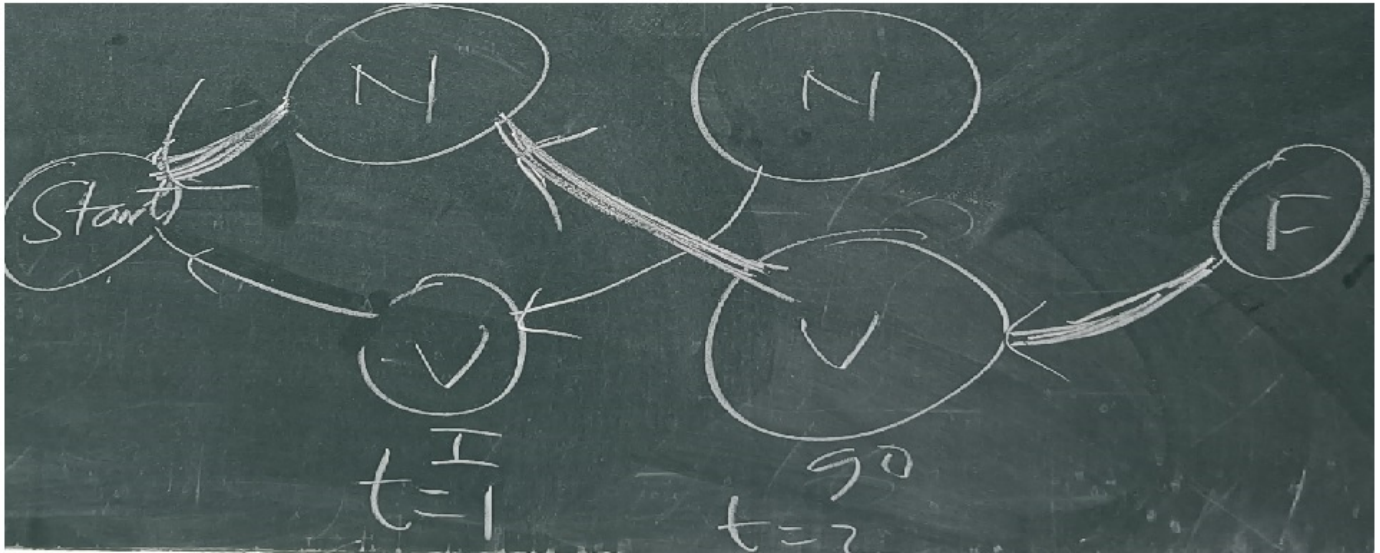
$$V_1(V) * a_{VV} * b_V(\text{go}) \}$$

$$= \max\{ 0.28 * 0.7 * 0.7,$$

$$0.09 * 0.2 * 0.7 \}$$

$$= 0.137$$

$$V_3(F) = \max\{ V_2(N) * a_{NE},$$
$$V_2(V) * a_{VE} \}$$
$$= 0.0137$$

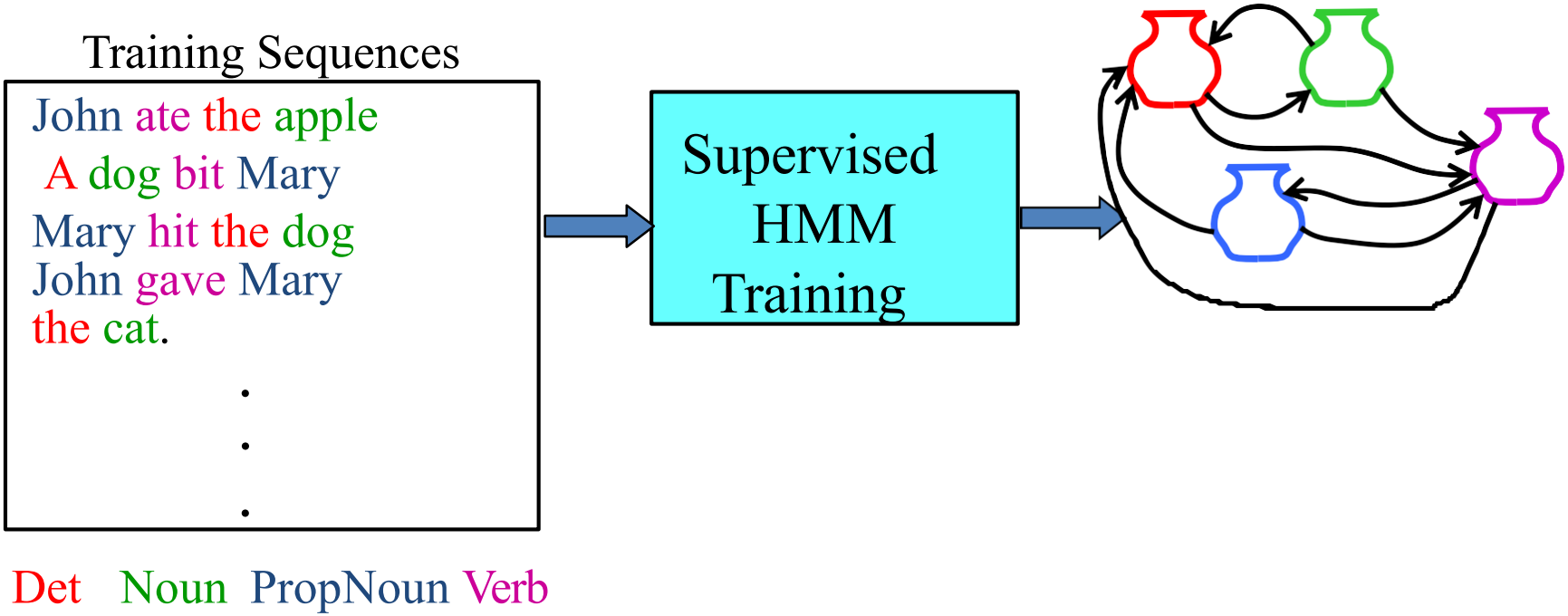


HMM Learning

- **Supervised Learning**: All training sequences are completely labeled (tagged).
- **Unsupervised Learning**: All training sequences are unlabelled (but generally know the number of tags,i.e. states).
- **Semisupervised Learning**: Some training sequences are labeled, most are unlabeled.

Supervised HMM Training

- If training sequences are labeled (tagged) with the underlying state sequences that generated them, then the parameters, $\lambda=\{A,B\}$ can all be estimated directly.



Supervised Parameter Estimation

- Estimate state transition probabilities based on tag bigram and unigram statistics in the labeled data.

$$a_{ij} = \frac{C(q_t = s_i, q_{t+1} = s_j)}{C(q_t = s_i)}$$

- Estimate the observation probabilities based on tag/word co-occurrence statistics in the labeled data.

$$b_j(k) = \frac{C(q_i = s_j, o_i = v_k)}{C(q_i = s_j)}$$

- Use appropriate smoothing if training data is sparse.

Learning and Using HMM Taggers

- Use a corpus of labeled sequence data to construct an HMM using supervised training.

- Given a novel unlabeled test sequence to tag, use the Viterbi algorithm to predict the most likely (globally optimal) tag sequence.

Evaluating Taggers

- Train on ***training set*** of labeled sequences.
- Possibly tune parameters based on performance on a ***development set***.
- Measure accuracy on a disjoint ***test set***.
- Generally measure ***tagging accuracy***, i.e. the percentage of tokens tagged correctly.
- Accuracy of most modern POS taggers, including HMMs is 96–97% (for Penn tagset trained on about 800K words) .
  - Generally matching human agreement level.

Simple Linear Chain CRF Features

- Modeling the conditional distribution is similar to that used in multinomial logistic regression.
- Create feature functions $f_k(Y_t, Y_{t-1}, X_t)$
  - Feature for each state transition pair i, j
    - $f_{i,j}(Y_t, Y_{t-1}, X_t) = 1$ if $Y_t = i$ and $Y_{t-1} = j$ and 0 otherwise
  - Feature for each state observation pair i, o
    - $f_{i,o}(Y_t, Y_{t-1}, X_t) = 1$ if $Y_t = i$ and $X_t = o$ and 0 otherwise
- **Note**: number of features grows quadratically in the number of states (i.e. tags).

Conditional Distribution for Linear Chain CRF

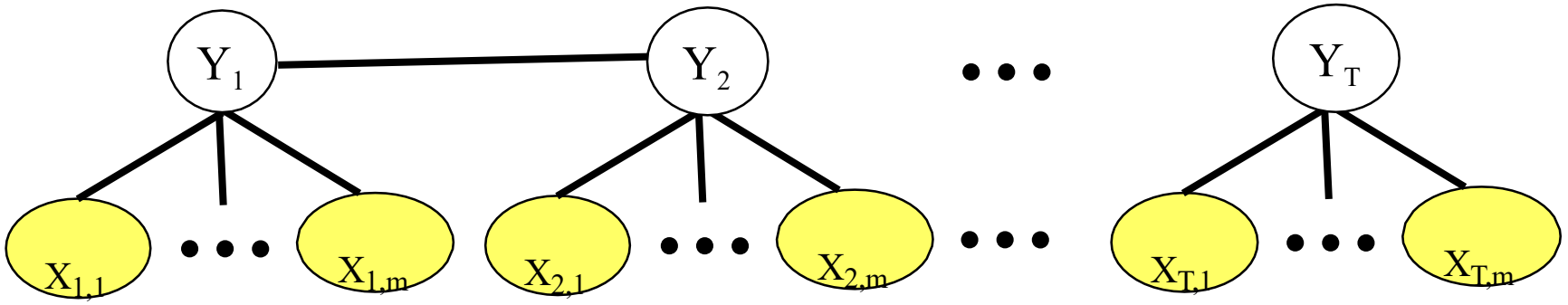
- Using these feature functions for a simple linear chain CRF, we can define:

$$P(Y \mid X) = \frac{1}{Z(X)} \exp\left(\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(Y_t, Y_{t-1}, X_t)\right)$$

$$Z(X) = \sum_{Y} \exp\left(\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(Y_t, Y_{t-1}, X_t)\right)$$

Adding Token Features to a CRF

- Can add token features $X_{i,j}$



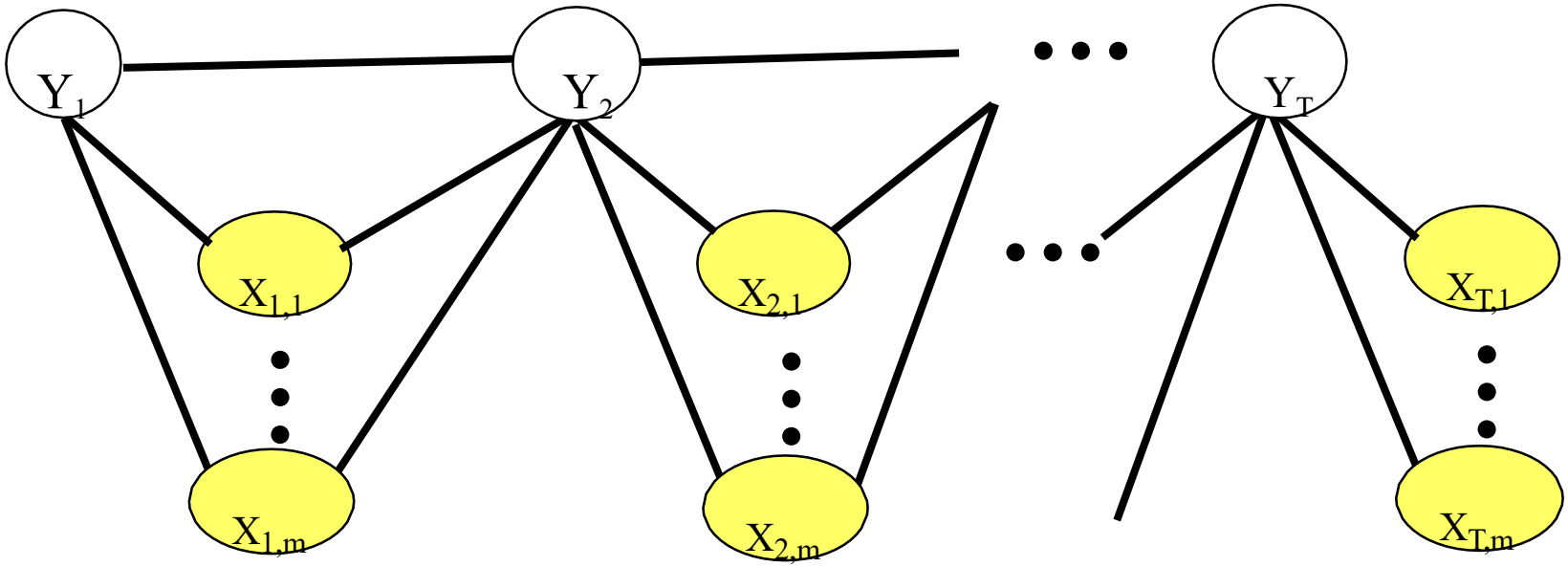
- Can add additional feature functions for each token feature to model conditional distribution.

Features in POS Tagging

- For POS Tagging, use lexicographic features of tokens.
  - Capitalized?
  - Start with numeral?
  - Ends in given suffix (e.g. "s", "ed", "ly")?

Enhanced Linear Chain CRF (standard approach)

- Can also condition transition on the current token features.



- Add feature functions:
  - $f_{i,j,k}(Y_t, Y_{t-1}, X)$ 1 if $Y_t = i$ and $Y_{t-1} = j$ and $X_{t-1,k} = 1$ and 0 otherwise

Supervised Learning (Parameter Estimation)

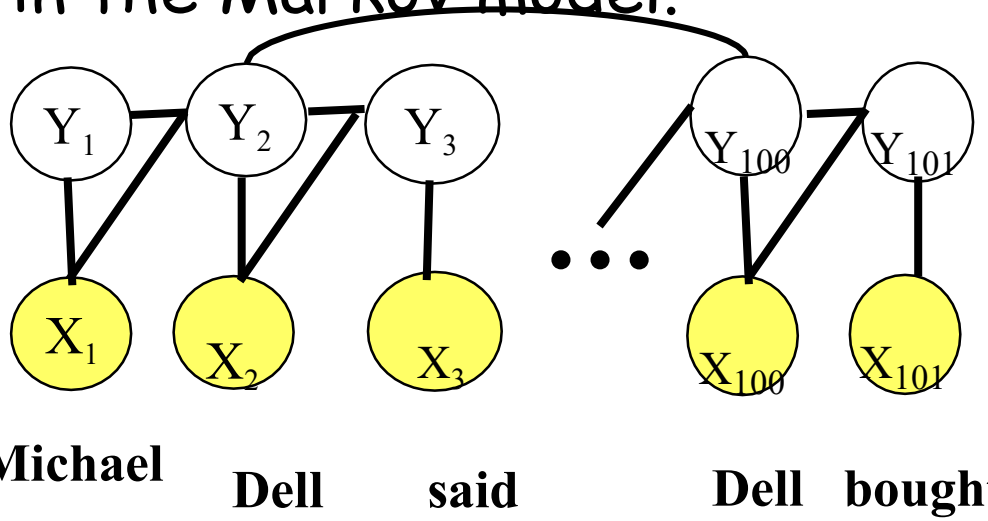
- As in logistic regression, use L-BFGS optimization procedure, to set λ weights to maximize CLL of the supervised training data.

Sequence Tagging (Inference)

- Variant of Viterbi algorithm can be used to efficiently, $O(TN^2)$, determine the globally most probable label sequence for a given token sequence using a given log-linear model of the conditional probability $P(Y \mid X)$.

Skip-Chain CRFs

- Can model some long-distance dependencies (i.e. the same word appearing in different parts of the text) by including long-distance edges in the Markov model.



- Additional links make exact inference intractable, so must resort to approximate inference to try to find the most probable labeling.

CRF Results

- Experimental results verify that they have superior accuracy on various sequence labeling tasks.
  - Part of Speech tagging
  - Noun phrase chunking
  - Named entity recognition
  - Semantic role labeling
- However, CRFs are much slower to train and do not scale as well to large amounts of training data.
  - Training for POS on full Penn Treebank (~1M words) currently takes "over a week."
- Skip-chain CRFs improve results.

State of the art - POS

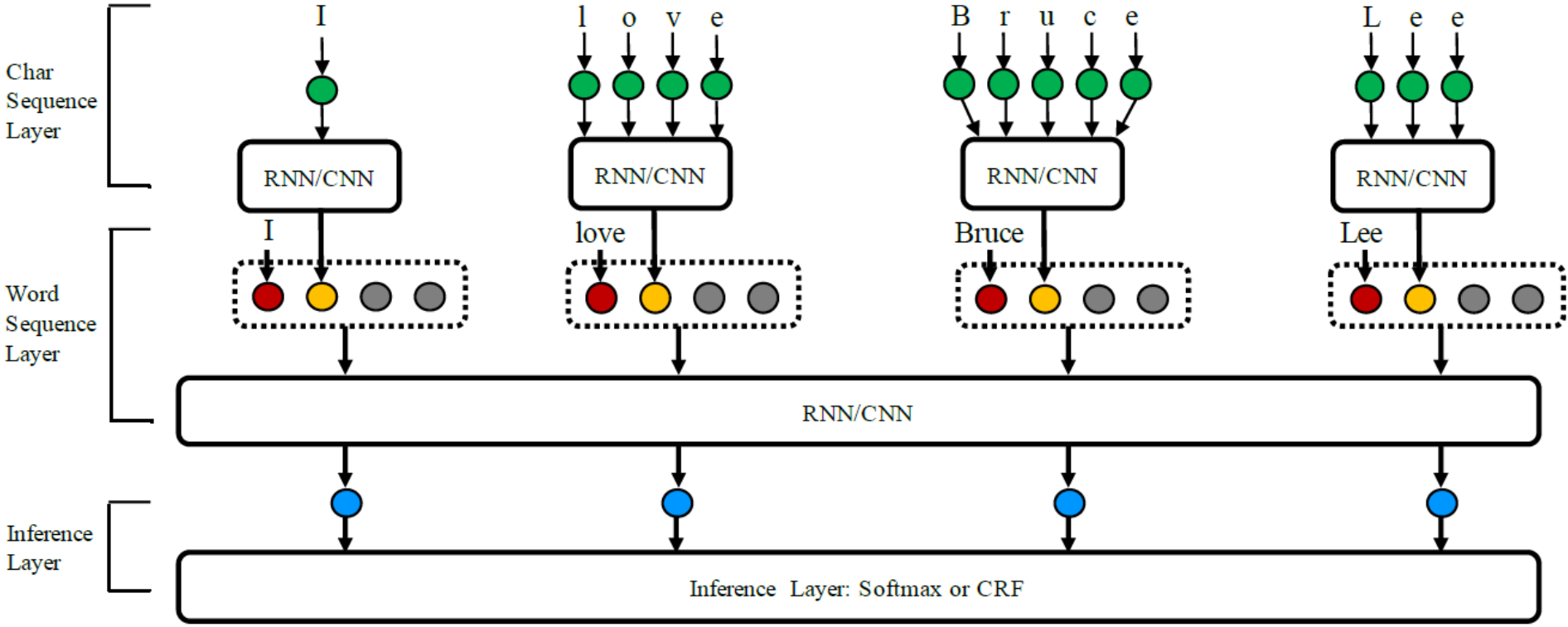
Penn Treebank

A standard dataset for POS tagging is the Wall Street Journal (WSJ) portion of the Penn Treebank, containing 45 different POS tags.

State of the art - POS

Model	Accuracy	Paper / Source	Code
Meta BiLSTM (Bohnet et al., 2018)	97.96	Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings	Official
Flair embeddings (Akbik et al., 2018)	97.85	Contextual String Embeddings for Sequence Labeling	Flair framework
Char Bi-LSTM (Ling et al., 2015)	97.78	Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation	
Adversarial Bi-LSTM (Yasunaga et al., 2018)	97.59	Robust Multilingual Part-of-Speech Tagging via Adversarial Training	
BiLSTM-CRF + IntNet (Xin et al., 2018)	97.58	Learning Better Internal Structure of Words for Sequence Labeling	
Yang et al. (2017)	97.55	Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks	
Ma and Hovy (2016)	97.55	End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF	
LM-LSTM-CRF (Liu et al., 2018)	97.53	Empowering Character-aware Sequence Labeling with Task-Aware Neural Language Model	
NCRF++ (Yang and Zhang, 2018)	97.49	NCRF++: An Open-source Neural Sequence Labeling Toolkit	NCRF++
Feed Forward (Vaswani et a. 2016)	97.4	Supertagging with LSTMs	
Bi-LSTM (Ling et al., 2017)	97.36	Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation	
Bi-LSTM (Plank et al., 2016)	97.22	Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss	

State of the art - POS



[NCRF++: An Open-source Neural Sequence Labeling Toolkit]

CRF Features - POS

```
def word2features(sent, i):
    word = sent[i][0]

    features = {
        'bias': 1.0,
        'word': word,
        'len(word)': len(word),
        'word[:4]': word[:4],
        'word[:3]': word[:3],
        'word[:2]': word[:2],
        'word[-3:]': word[-3:],
        'word[-2:]': word[-2:],
        'word[-4:]': word[-4:],
        'word.lower()': word.lower(),
        'word.stemmed': re.sub(r'(.{2,}?)([aeiougyn]+$)',r'\1',
word.lower()),
        'word.ispunctuation': (word in string.punctuation),
```

CRF Features - POS

```
 'word.isdigit()': word.isdigit(),
    }    if i > 0:
        word1 = sent[i-1][0]
        features.update({
            '-1:word': word1,
            '-1:len(word)': len(word1),
            '-1:word.lower()': word1.lower(),
            '-1:word.stemmed': re.sub(r'(.{2,}?)([aeiougyn]+
$)',r'\1', word1.lower()),
            '-1:word[:3]': word1[:3],
            '-1:word[:2]': word1[:2],
            '-1:word[-3:]': word1[-3:],
            '-1:word[-2:]': word1[-2:],
            '-1:word.isdigit()': word1.isdigit(),
            '-1:word.ispunctuation': (word1 in
string.punctuation),
        })     else:
        features['BOS'] = True
```

CRF Features - POS

```
if i > 1:
        word2 = sent[i-2][0]
        features.update({
            '-2:word': word2,
            '-2:len(word)': len(word2),
            '-2:word.lower()': word2.lower(),
            '-2:word[:3]': word2[:3],
            '-2:word[:2]': word2[:2],
            '-2:word[-3:]': word2[-3:],
            '-2:word[-2:]': word2[-2:],
            '-2:word.isdigit()': word2.isdigit(),
            '-2:word.ispunctuation': (word2 in
string.punctuation),
        })

    if i < len(sent)-1:
        word1 = sent[i+1][0]
        features.update({
            '+1:word': word1,
```

CRF Features - POS

```
'+1:len(word)': len(word1),
            '+1:word.lower()': word1.lower(),
            '+1:word[:3]': word1[:3],
            '+1:word[:2]': word1[:2],
            '+1:word[-3:]': word1[-3:],
            '+1:word[-2:]': word1[-2:],
            '+1:word.isdigit()': word1.isdigit(),
            '+1:word.ispunctuation': (word1 in
string.punctuation),
        })

    else:
        features['EOS'] = True    if i < len(sent) - 2:
        word2 = sent[i+2][0]
        features.update({
            '+2:word': word2,
            '+2:len(word)': len(word2),
            '+2:word.lower()': word2.lower(),
```

CRF Features - POS

```
'+2:word.stemmed': re.sub(r'(.{2,}?)([aeiougyn]+$)',r'\1',
word2.lower()),
            '+2:word[:3]': word2[:3],
            '+2:word[:2]': word2[:2],
            '+2:word[-3:]': word2[-3:],
            '+2:word[-2:]': word2[-2:],
            '+2:word.isdigit()': word2.isdigit(),
            '+2:word.ispunctuation': (word2 in
string.punctuation),
        })

    return features
```