Sequence Modelling
HMM

Sudeshna Sarkar

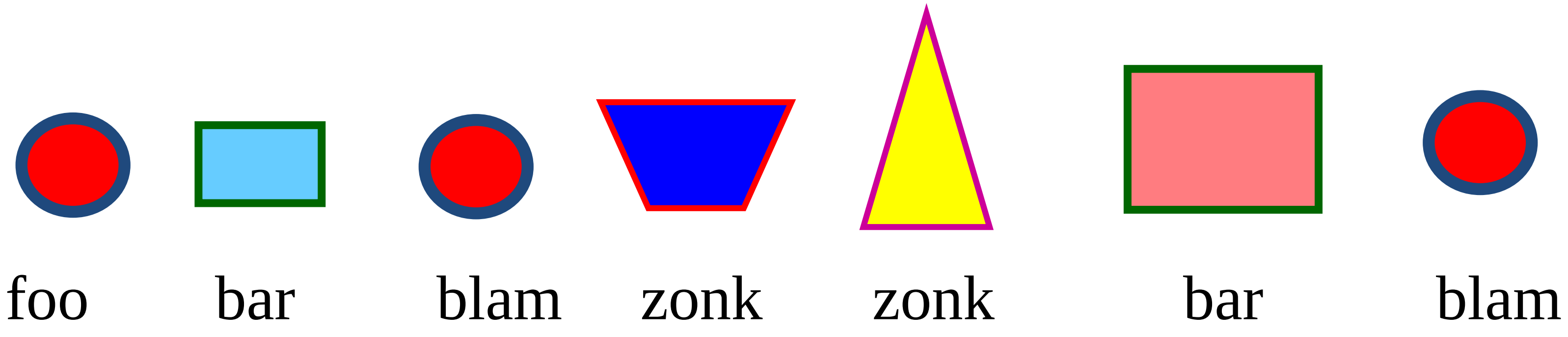
1 August 2019

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)

Sequence Labeling Problem

- Many NLP problems can viewed as sequence labeling.
- Each token in a sequence is assigned a label.
- Labels of tokens are dependent on the labels of other tokens in the sequence, particularly their neighbors (not i.i.d).



Information Extraction

- Identify phrases in language that refer to specific types of entities and relations in text.
- Named entity recognition is task of identifying names of people, places, organizations, etc. in text.

  people organizations places

  - Michael Dell is the CEO of Dell Computer Corporation and lives in Austin Texas.
- Extract pieces of information relevant to a specific application, e.g. used car ads:

  make model year mileage price

  - For sale, 2002 Toyota Prius, 20,000 mi, $15K or best offer. Available starting July 30, 2006.

Semantic Role Labeling

- For each clause, determine the semantic role played by each noun phrase that is an argument to the verb.

  agent patient source destination instrument

  - John drove Mary from Austin to Dallas in his Toyota Prius.
  - The hammer broke the window.
- Also referred to a "case role analysis," "thematic analysis," and "shallow semantic parsing"

Bioinformatics

- Sequence labeling also valuable in labeling genetic sequences in genome analysis.

  extron intron

  - AGCTAACGTTCGATACGGATTACAGCCT

Problems with Sequence Labeling as Classification

- Need to integrate information from category of tokens on both sides.
- Difficult to propagate uncertainty between decisions and “collectively” determine the most likely joint assignment of categories to all of the tokens in a sequence.

Probabilistic Sequence Models

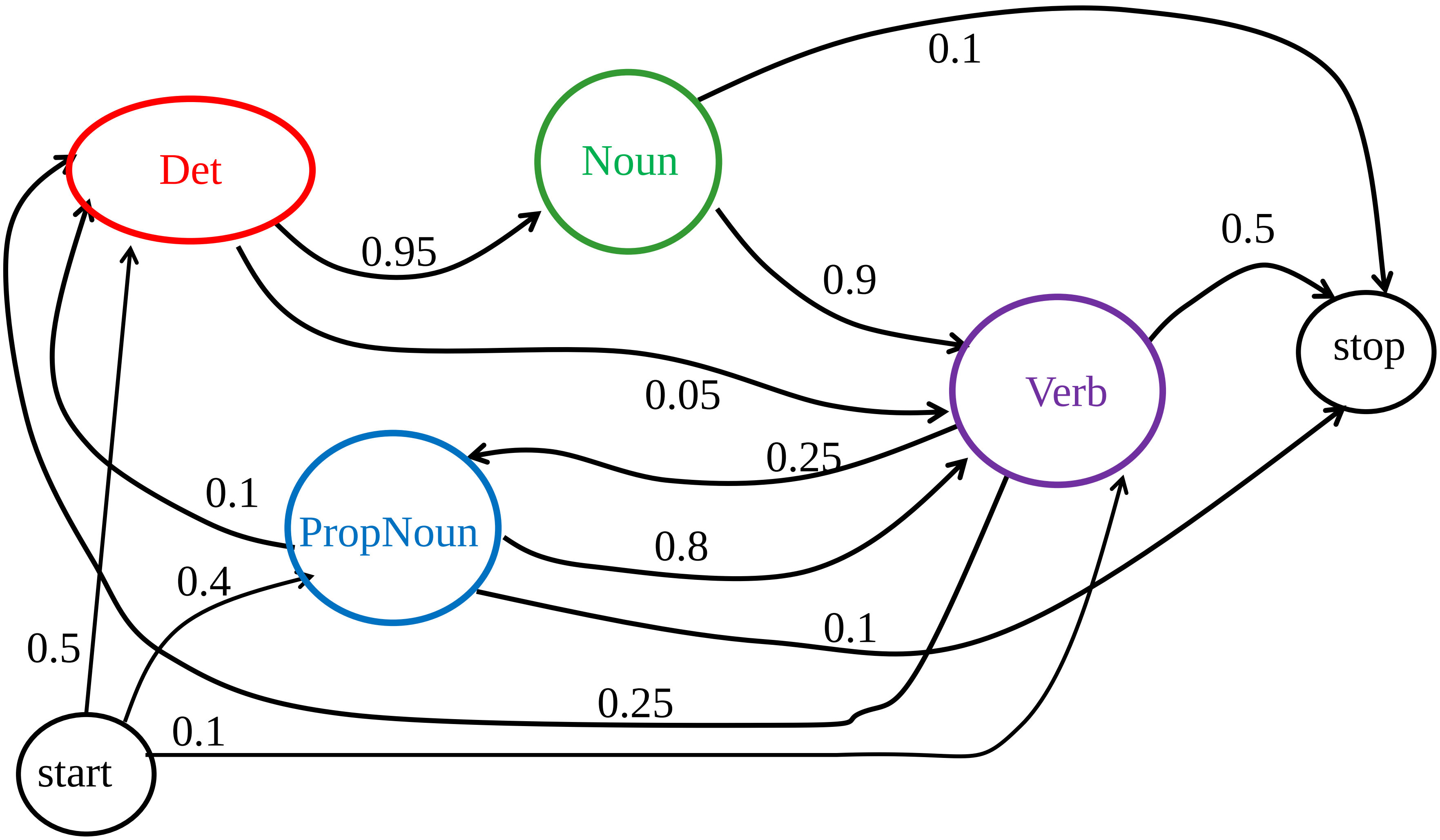
- Probabilistic sequence models allow integrating uncertainty over multiple, interdependent classifications and collectively determine the most likely global assignment.

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

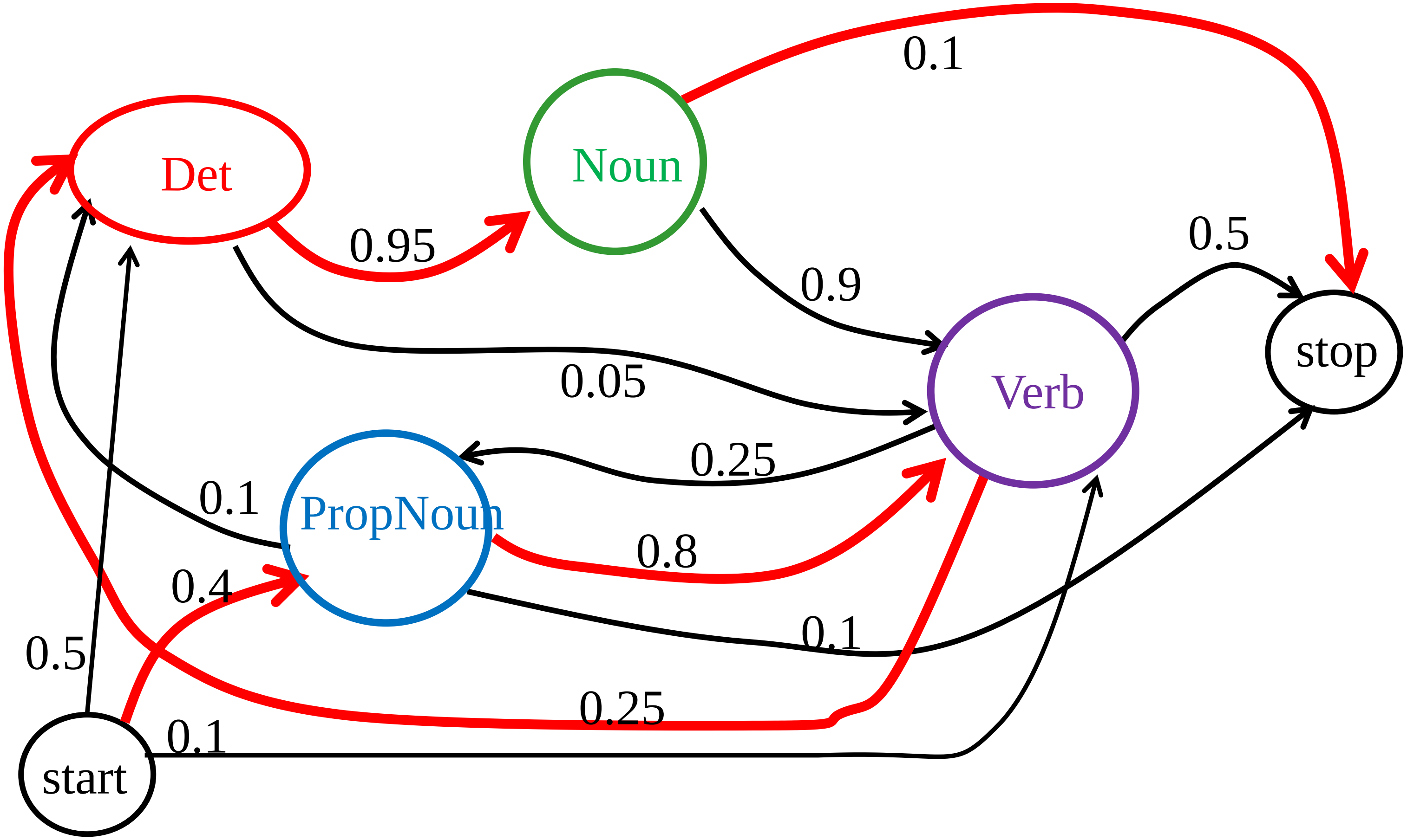
Markov Model / Markov Chain

- 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

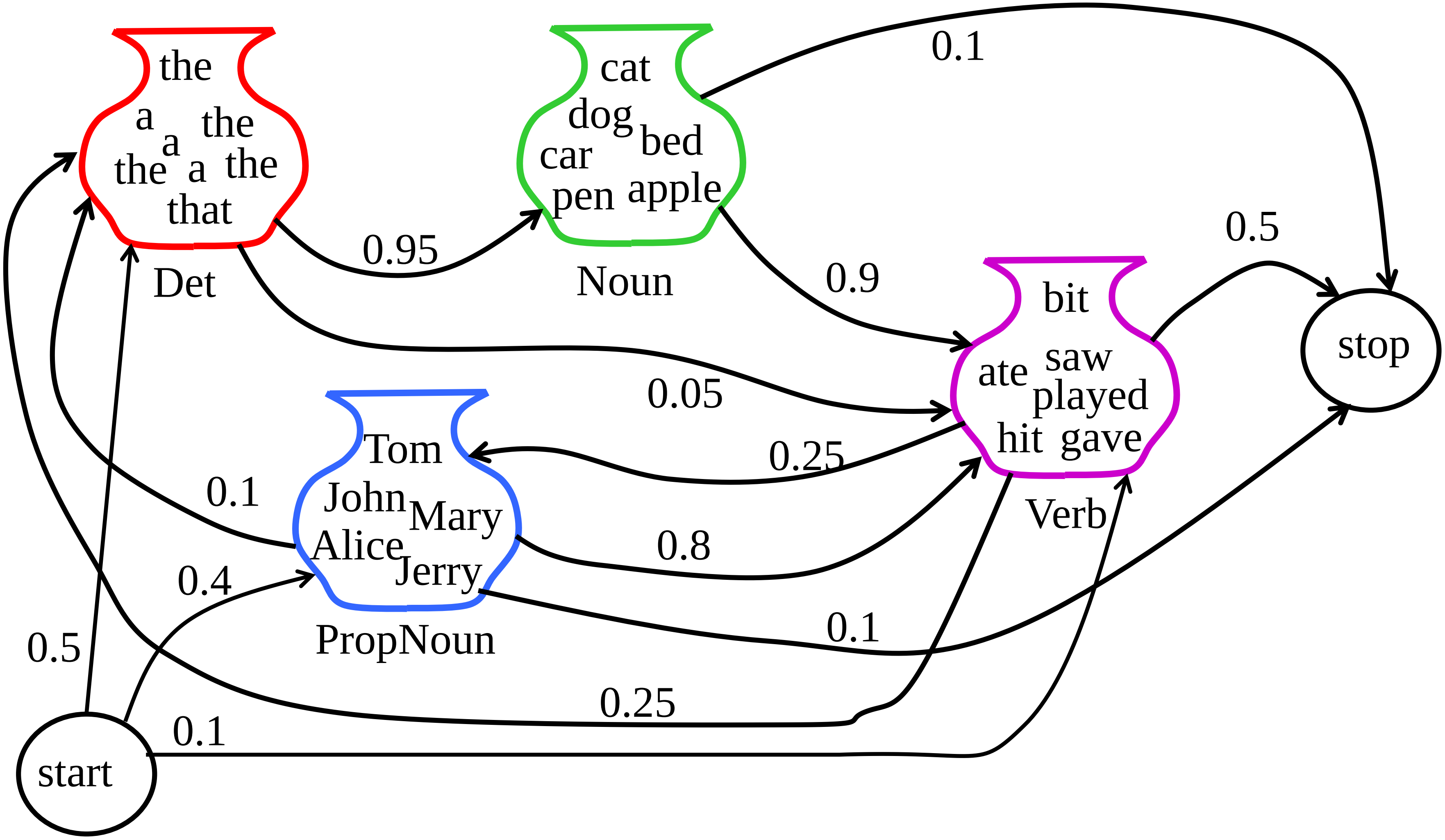


P(PropNoun Verb Det Noun) = 0.4∗0.8∗0.25∗0.95∗0.1=0.0076

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* $B=\{b_j(k)\}$

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

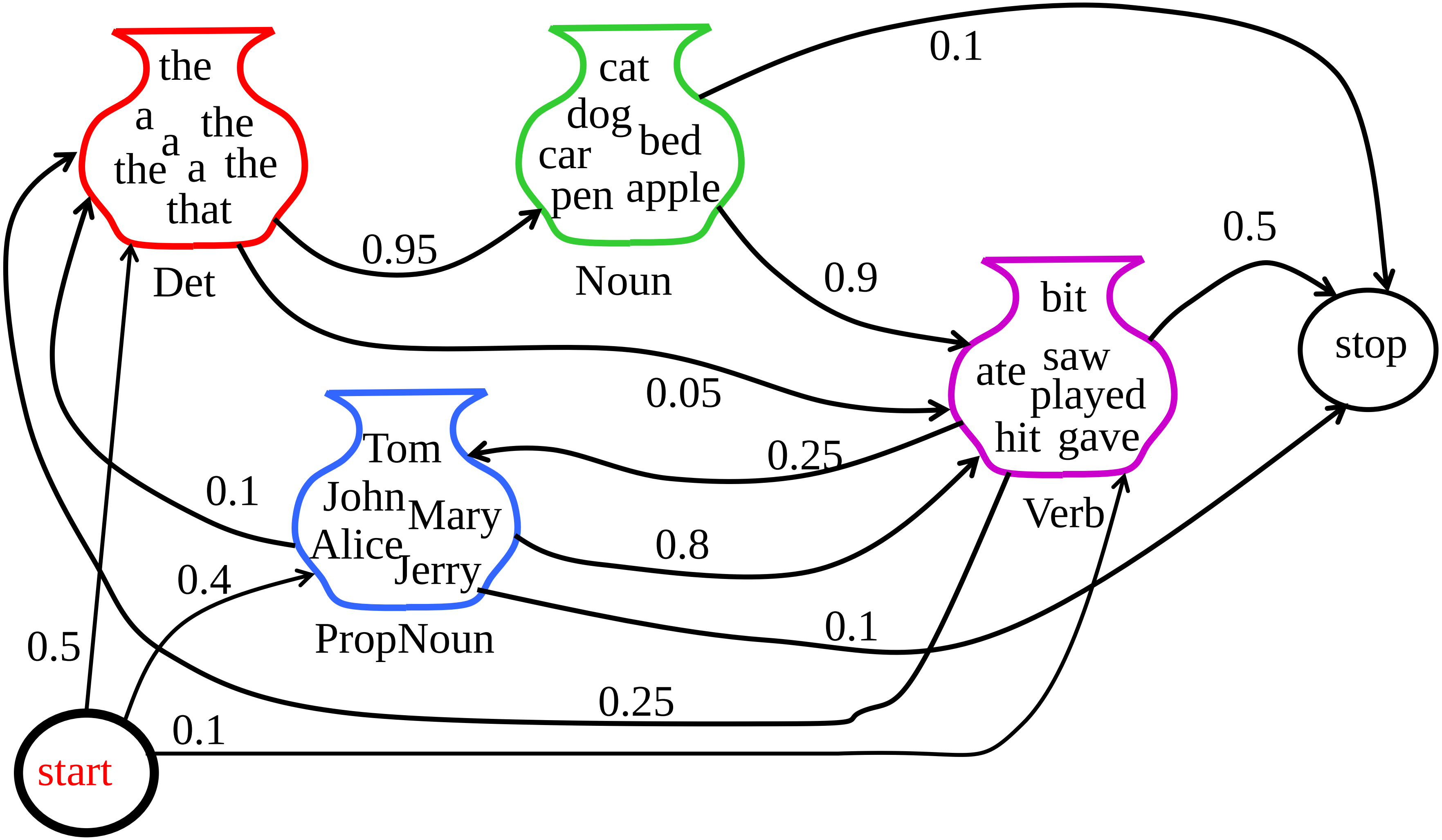
- Total parameter set $\lambda=\{A,B\}$

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?

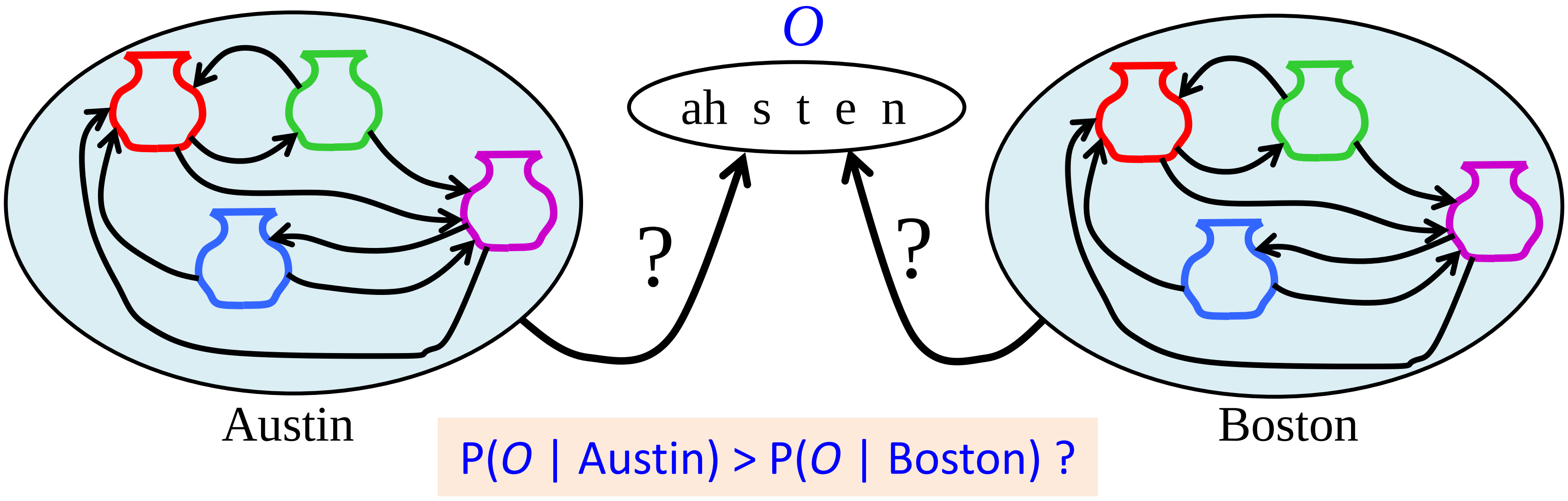
HMM: Observation Likelihood

- 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| λ) ?

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

Sequence Classification

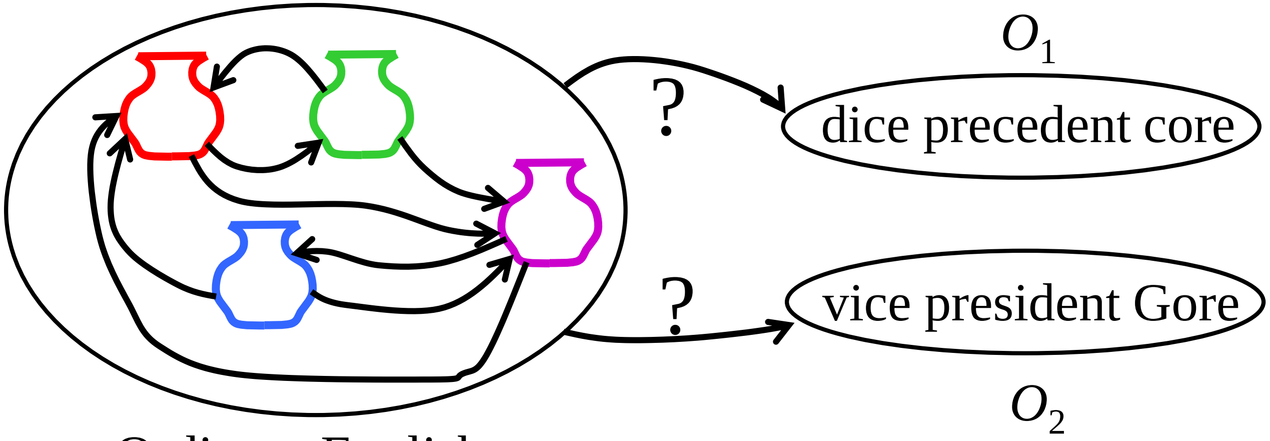
- Assume an HMM is available for each category (i.e. language).
- What is the most likely category for a given observation sequence, i.e. which category's HMM is most likely to have generated it?
- Used in speech recognition to find most likely word model to have generate a given sound or phoneme sequence.



P(*O* | Austin) > P(*O* | Boston) ?

Most Likely Sequence

- Of two or more possible sequences, which one was most likely generated by a given model?
- Used to score alternative word sequence interpretations in speech recognition.



O_1

?

dice precedent core

?

vice president Gore

O_2

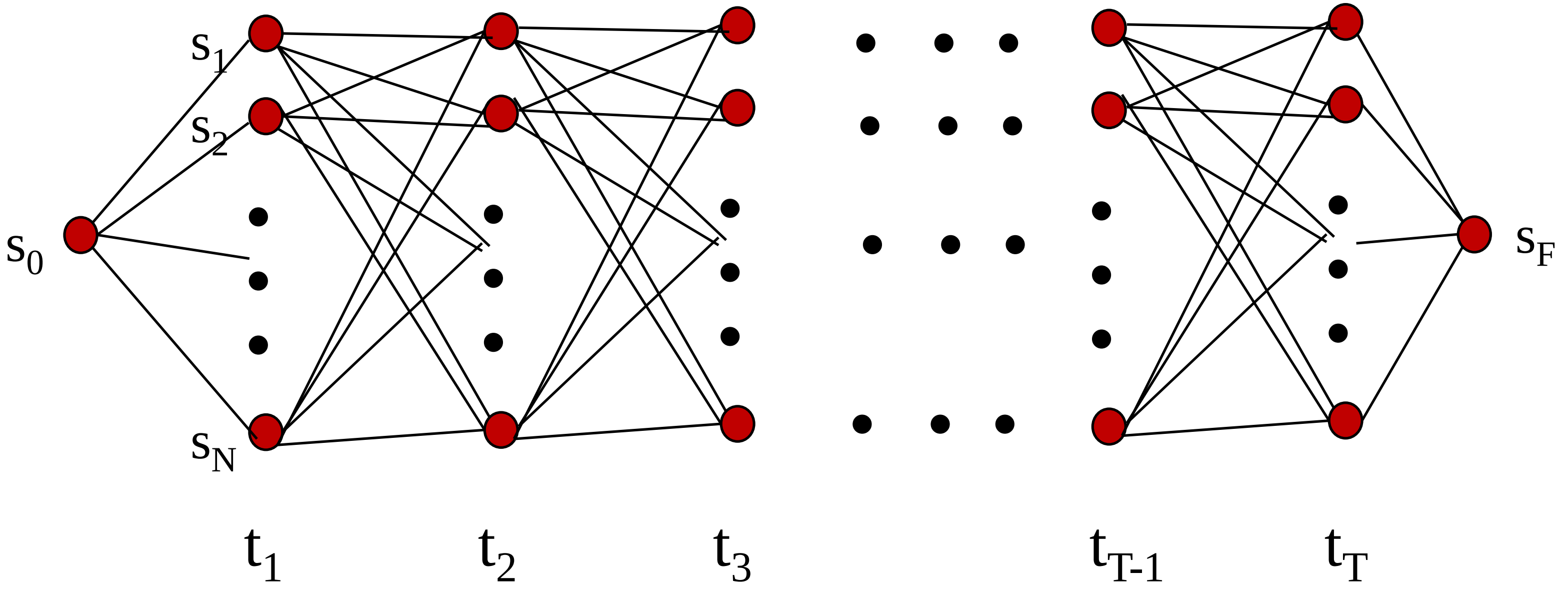
Ordinary English

P(O_2 | OrdEnglish) > P(O_1 | OrdEnglish) ?

HMM: Observation Likelihood Efficient Solution

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

Forward Trellis



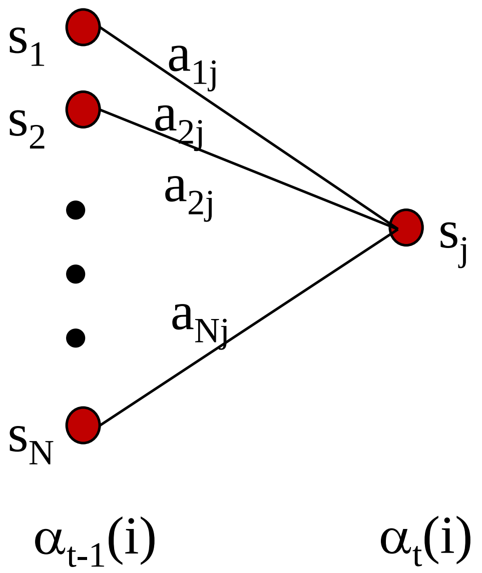
Continue forward in time until reaching final time point and sum probability of ending in final state.

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*).

$$\boxed{\alpha_t(j) = P(o_1, o_2, ..o_t,\ q_t = s_j \mid \lambda)}$$

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 the Forward Probabilities

- Initialization

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

- Recursion

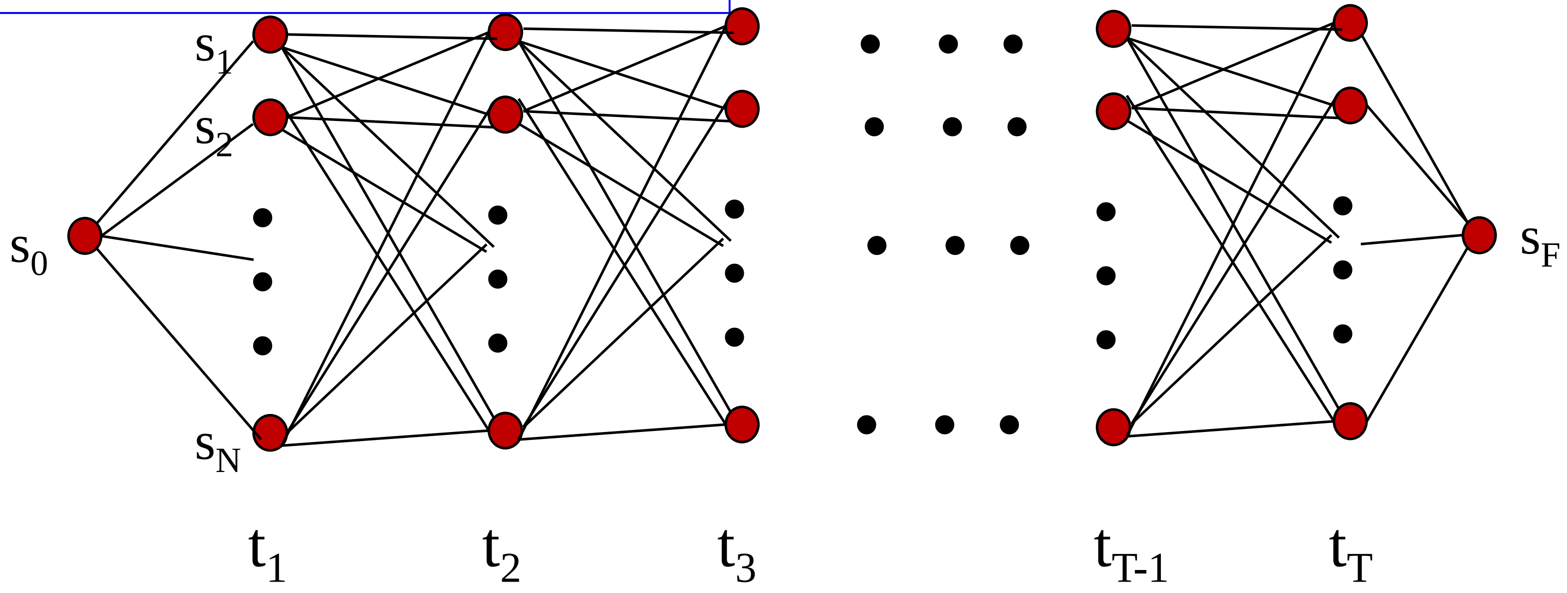
$$\boxed{\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 1 \le j \le N, \;\; 1 < t \le T}$$

- Termination

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

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

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



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

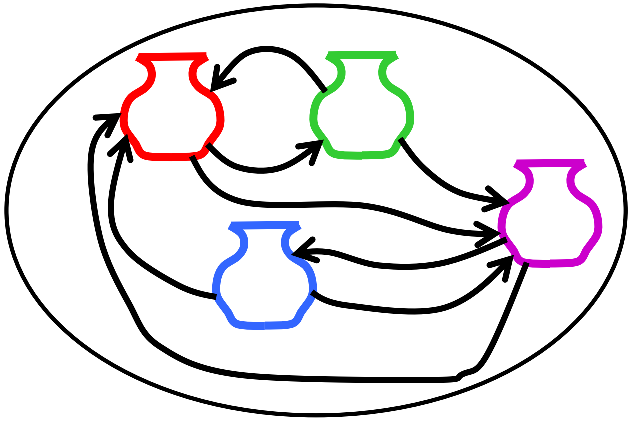
$$P(O \mid \lambda) = \alpha_{T+1}(s_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

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)

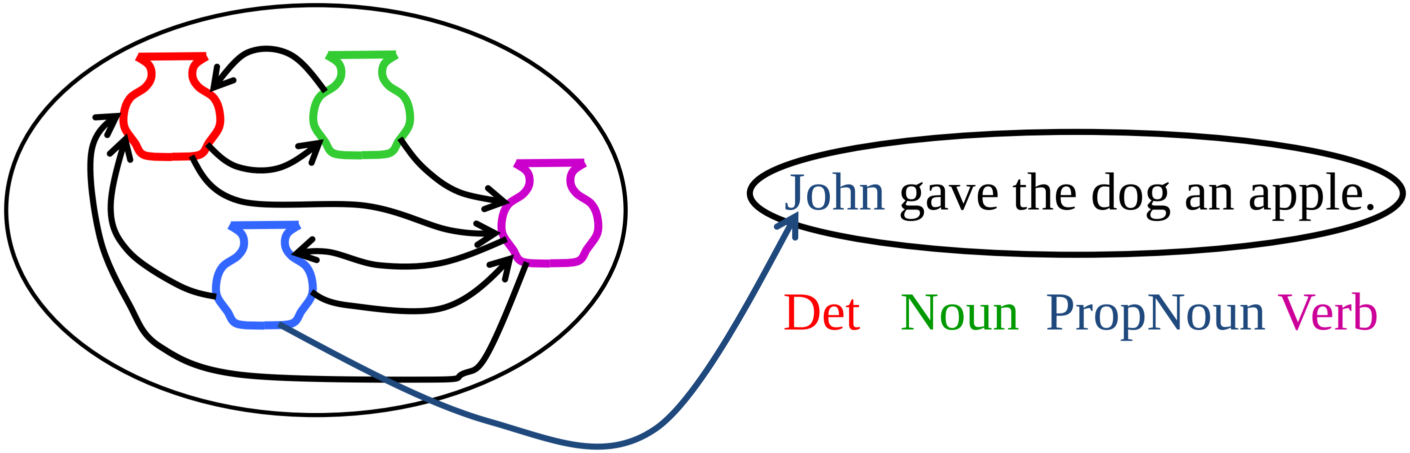
- Given an observation sequence, *O*, and a model, λ, what is ***the most likely state sequence***, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?



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?

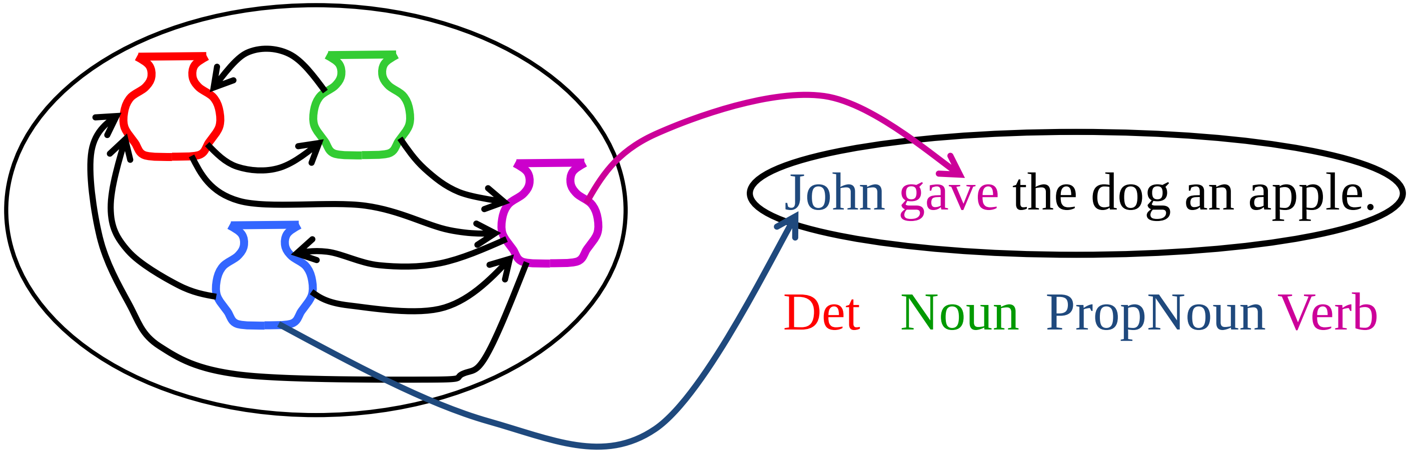


John gave the dog an apple.

Det Noun PropNoun Verb

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?

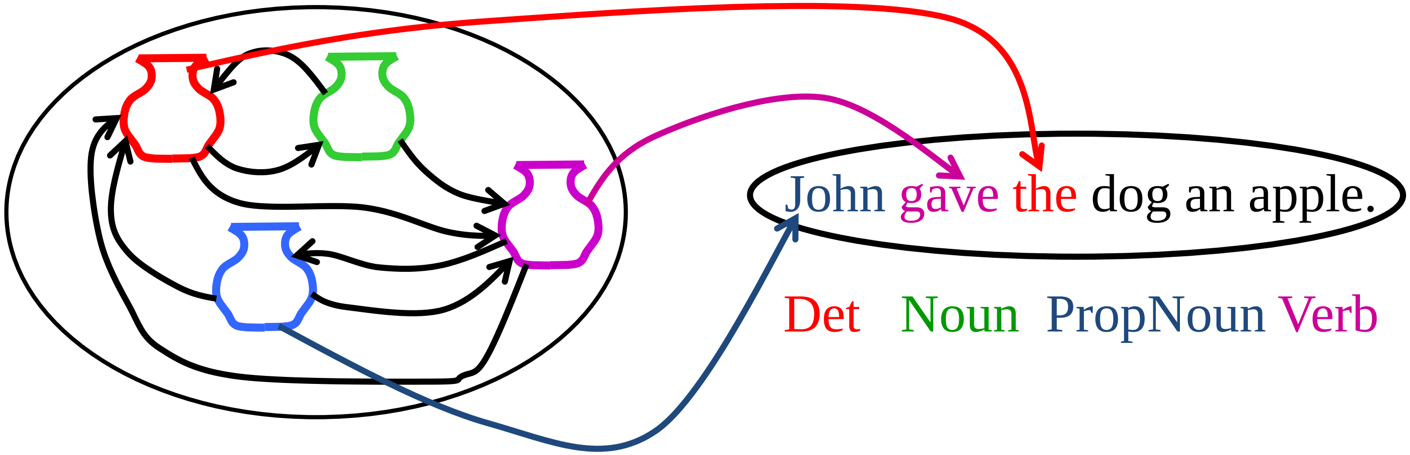


John gave the dog an apple.

Det Noun PropNoun Verb

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?

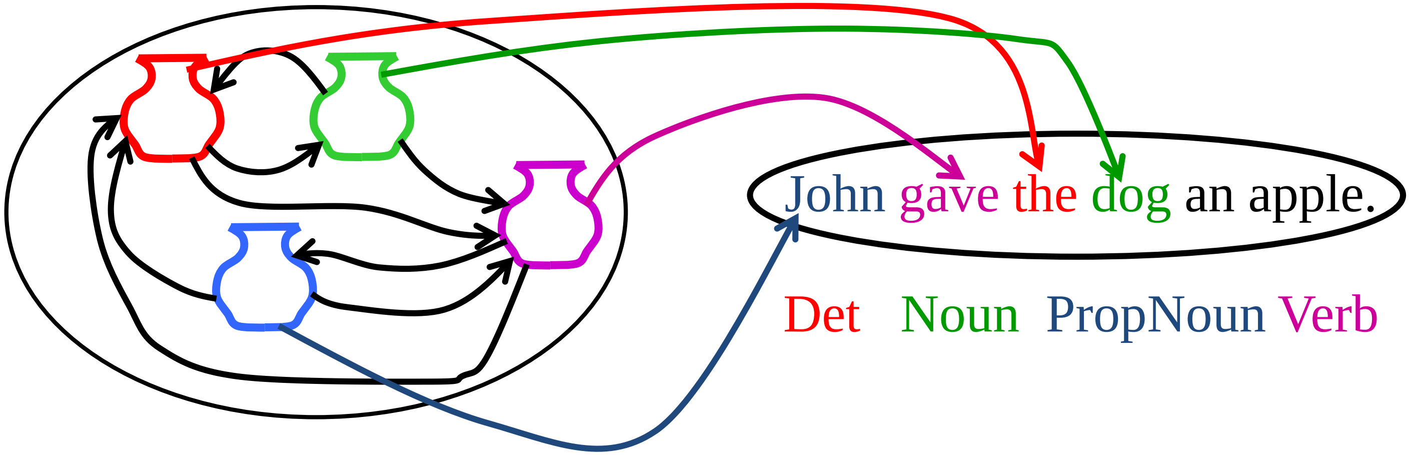


John gave the dog an apple.

Det Noun PropNoun Verb

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?

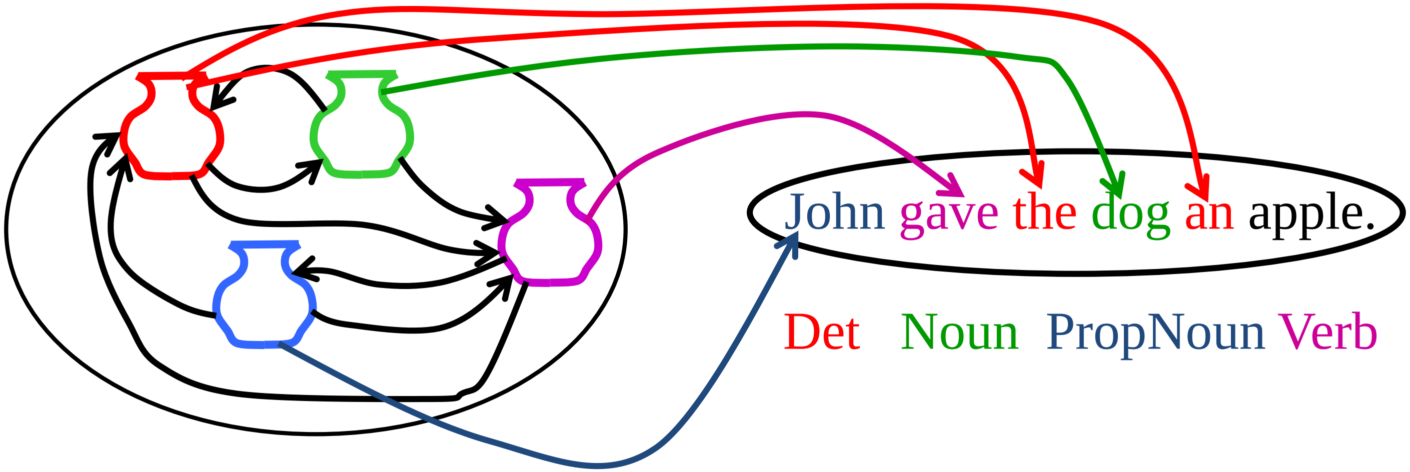


John gave the dog an apple.

Det Noun PropNoun Verb

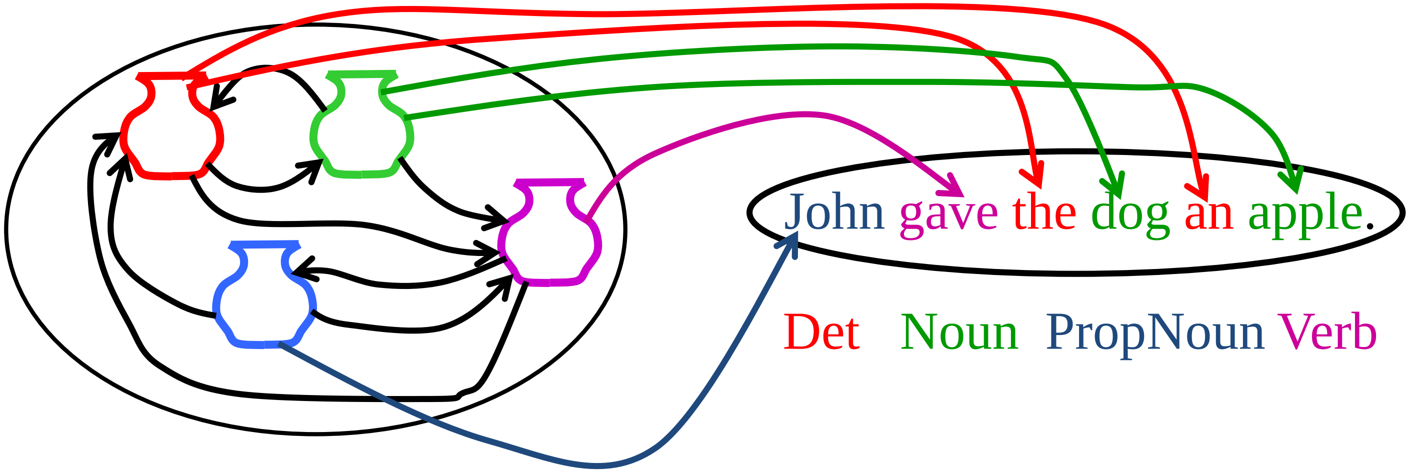
Most Likely State Sequence

- 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

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

- 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

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

- Recursion

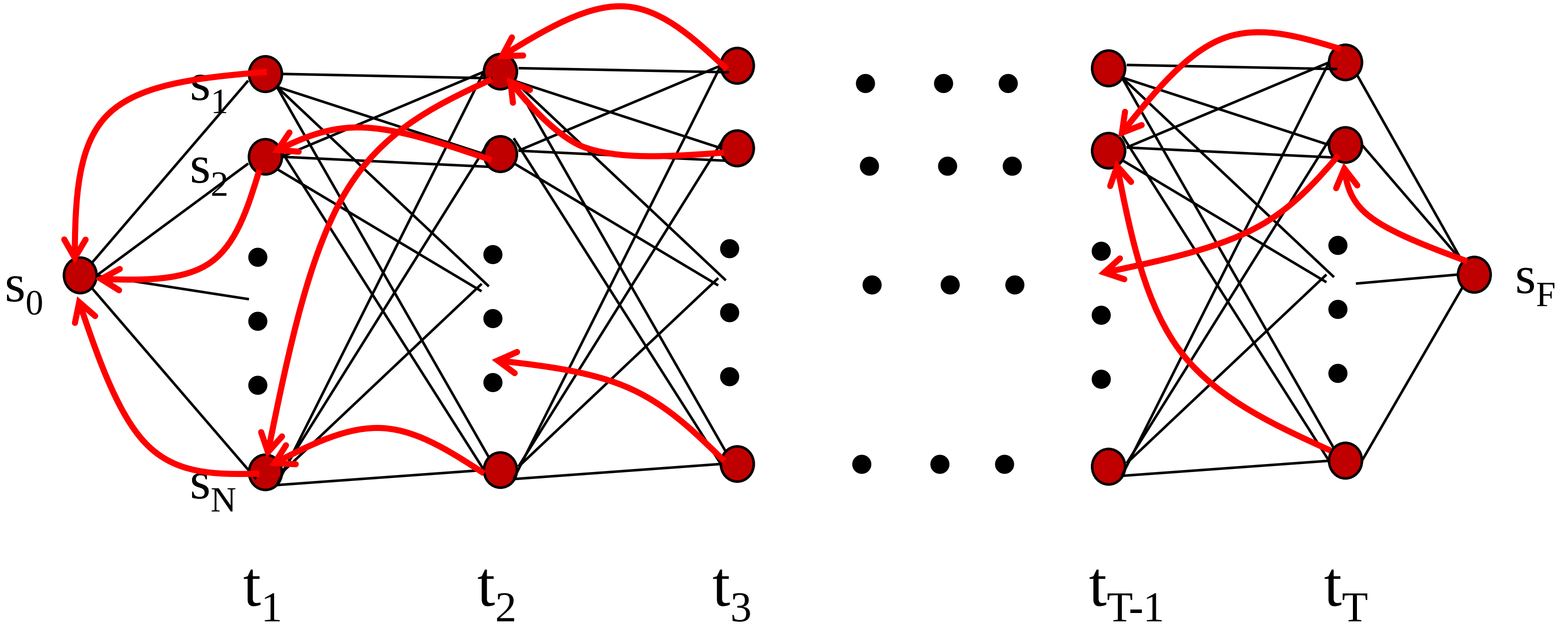
$$\boxed{bt_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, \;\; 1 \le t \le T}$$

- Termination

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

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

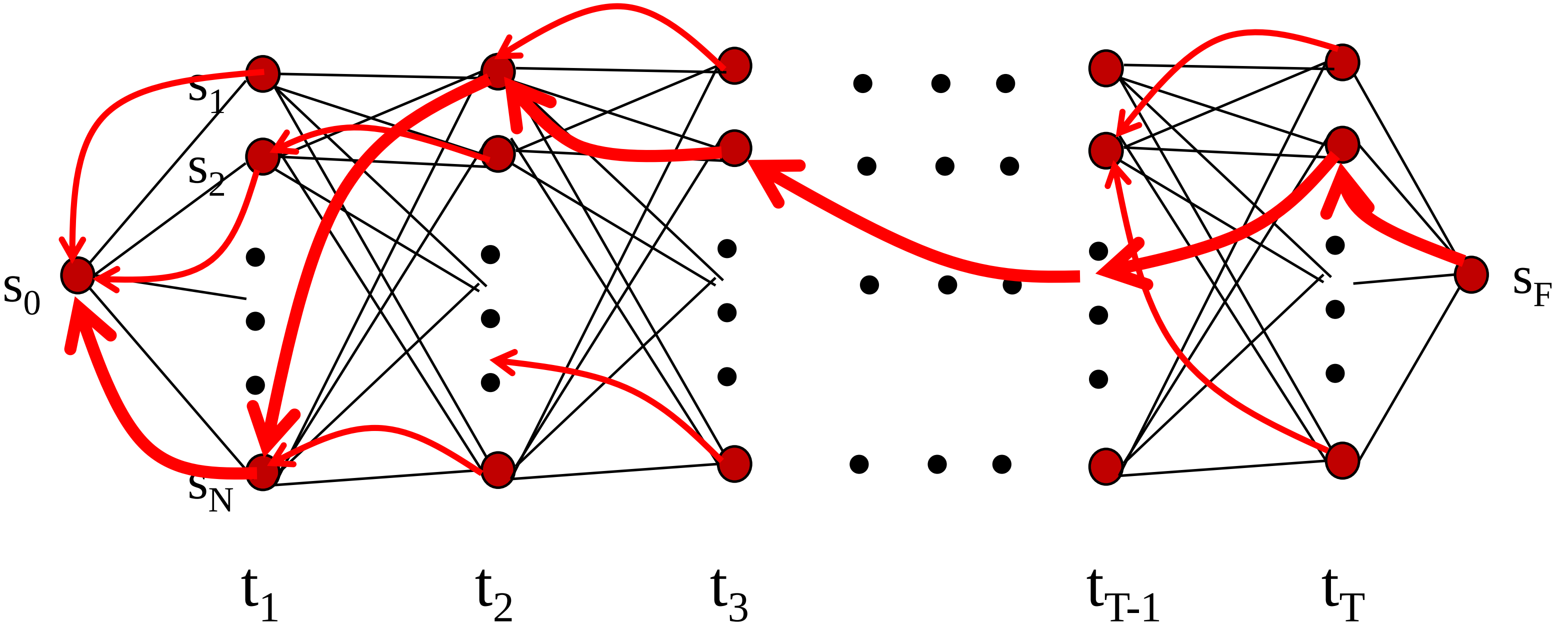
Viterbi Backpointers



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

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

Viterbi Backtrace



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

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

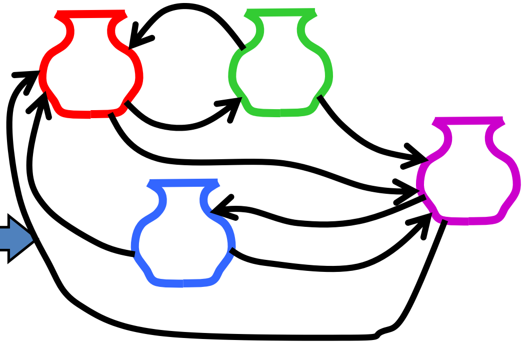
- If training sequences are labeled (tagged) with the underlying state sequences that generated them, then the parameters, λ={A,B} can all be estimated directly.

Training Sequences

John ate the apple
A dog bit Mary
Mary hit the dog
John gave Mary the cat.
.
.
.

Det Noun PropNoun Verb

Supervised HMM Training



Supervised Parameter Estimation

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

$$\boxed{a_{ij} = \frac{C(q_t = s_i, \mathrm{q}_{\mathrm{t+1}} = s_j)}{C(q_t = s_i)}}$$

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

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

- 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.

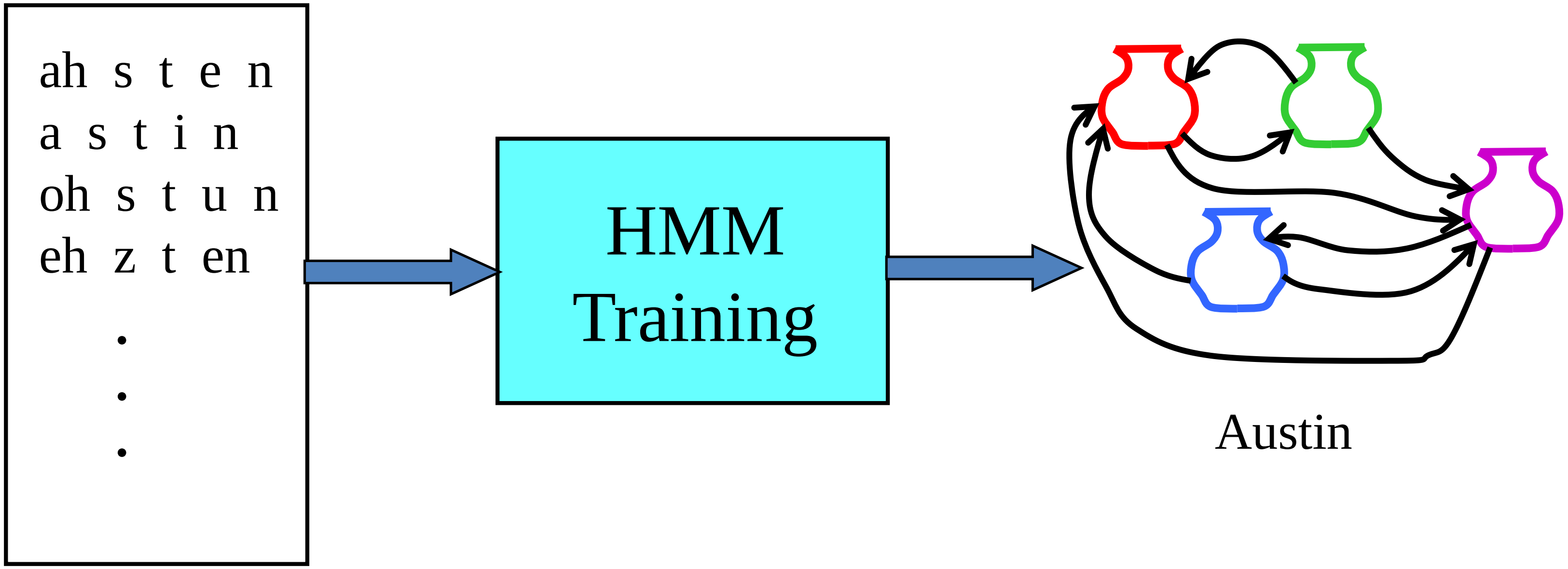
- Given a novel unlabeled test sequence to tag, use the Viterbi algorithm to predict the most likely (globally optimal) tag sequence.

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.

Unsupervised
Maximum Likelihood Training

Training Sequences



ah s t e n

a s t i n

oh s t u n

eh z t en

.

.

.

HMM Training

Austin

Maximum Likelihood Training

- Given an observation sequence, *O,*
  what set of parameters, λ, for a given model maximizes the probability that this data was generated from this model (P(*O*| λ))?

- Used to train an HMM model and properly induce its parameters from a set of training data.

Bayes Theorem

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Simple proof from definition of conditional probability:

$$P(H \mid E) = \frac{P(H \wedge E)}{P(E)} \quad \text{(Def. cond. prob.)}$$

$$P(E \mid H) = \frac{P(H \wedge E)}{P(H)} \quad \text{(Def. cond. prob.)}$$

$$P(H \wedge E) = P(E \mid H)P(H)$$

QED: $$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Maximum Likelihood vs. Maximum A Posteriori (MAP)

- The MAP parameter estimate is the most likely given the observed data, *O*.

$$\boxed{\lambda_{MAP} = \underset{\lambda}{\operatorname{argmax}} P(\lambda \mid O) = \underset{\lambda}{\operatorname{argmax}} \frac{P(O \mid \lambda) P(\lambda)}{P(O)}}$$

$$\boxed{\underset{\lambda}{\operatorname{argmax}} P(O \mid \lambda) P(\lambda)}$$

  - If all parameterizations are assumed to be equally likely *a priori*, then MLE and MAP are the same.
  - If parameters are given priors (e.g. Gaussian or Laplacian with zero mean), then MAP is a principled way to perform smoothing or regularization.

HMM: Maximum Likelihood Training Efficient Solution

- There is no known efficient algorithm for finding the parameters, λ, that truly maximizes P(*O*| λ).
- However, using iterative re-estimation, the Baum-Welch algorithm (a.k.a. forward-backward) , a version of a standard statistical procedure called Expectation Maximization (EM), is able to *locally* maximize P(*O*| λ).
- In practice, EM is able to find a good set of parameters that provide a good fit to the training data in many cases.

EM Algorithm

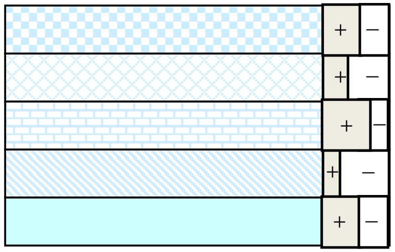
- Iterative method for learning probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories.
- Learn an initial probabilistic model by estimating model parameters θ from this randomly labeled data.
- Iterate following two steps until convergence:
  - Expectation (E-step): Compute $P(c_i \mid E)$ for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
  - Maximization (M-step): Re-estimate the model parameters, θ, from the probabilistically re-labeled data.

EM

Initialize:

Assign random probabilistic labels to unlabeled data

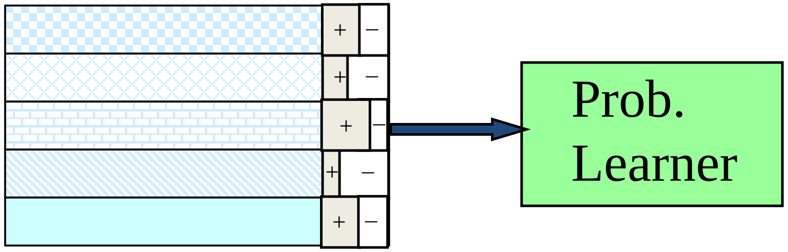
Unlabeled Examples



EM

Initialize:

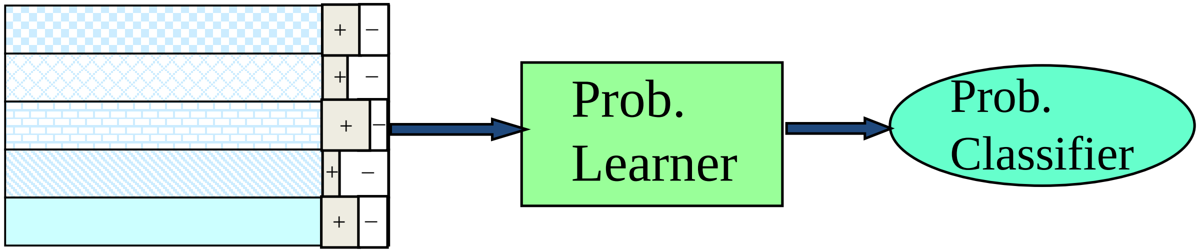
Give soft-labeled training data to a probabilistic learner



EM

Initialize:

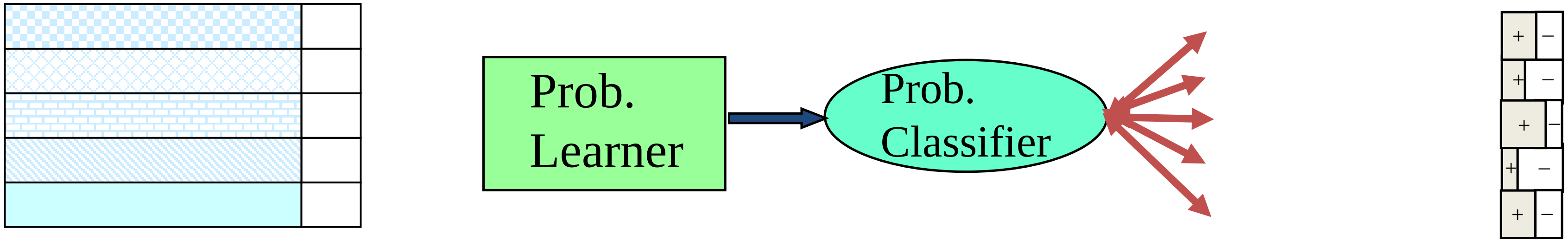
Produce a probabilistic classifier



EM

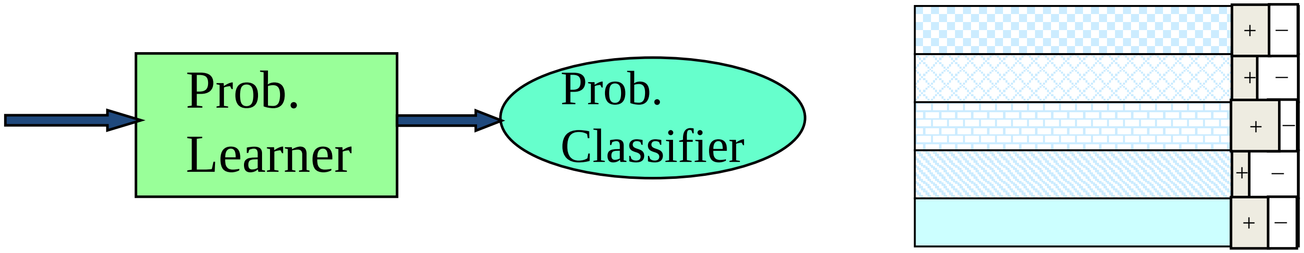
E Step:

Relabel unlabeled data using the trained classifier



M step:

Retrain classifier on relabeled data



Continue EM iterations until probabilistic labels on unlabeled data converge.

Sketch of Baum-Welch (EM) Algorithm for Training HMMs

Assume an HMM with *N* states.

Randomly set its parameters λ=(*A*,*B*)

(making sure they represent legal distributions)

Until converge (i.e. λ no longer changes) do:

E Step: Use the forward/backward procedure to determine the probability of various possible state sequences for generating the training data

M Step: Use these probability estimates to re-estimate values for all of the parameters λ

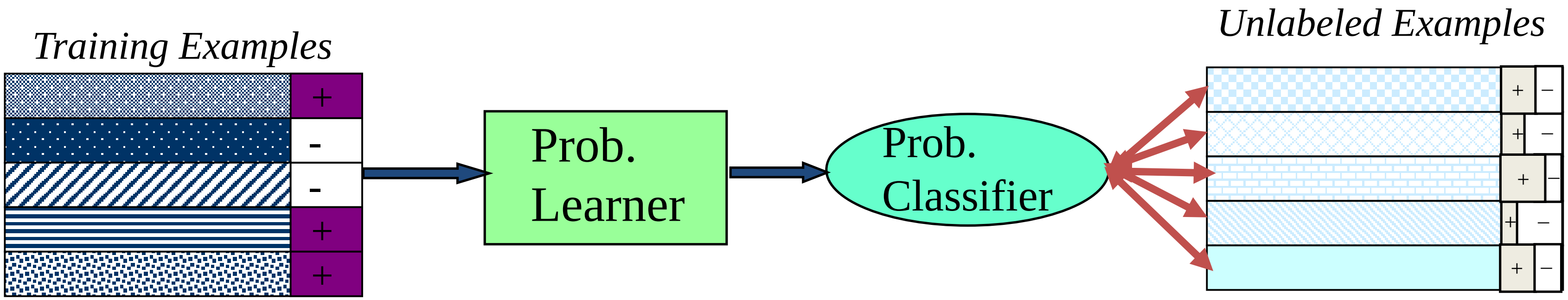
EM Properties

- Each iteration changes the parameters in a way that is guaranteed to increase the likelihood of the data: $P(O|\lambda)$.
- Anytime algorithm: Can stop at any time prior to convergence to get approximate solution.
- Converges to a local maximum.

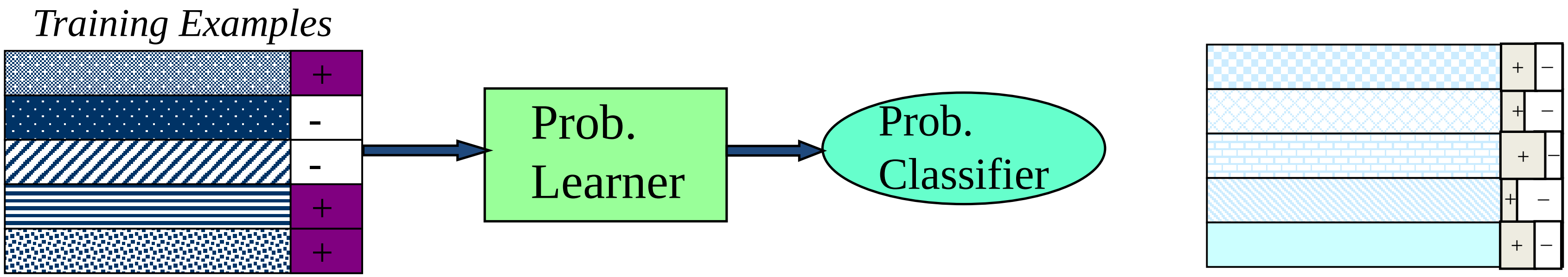
Semi-Supervised Learning

- EM algorithms can be trained with a mix of labeled and unlabeled data.
- EM basically predicts a probabilistic (soft) labeling of the instances and then iteratively retrains using supervised learning on these predicted labels (“self training”).
- EM can also exploit supervised data:
  - 1) Use supervised learning on labeled data to initialize the parameters (instead of initializing them randomly).
  - 2) Use known labels for supervised data instead of predicting soft labels for these examples during retraining iterations.

Semi-Supervised EM



Semi-Supervised EM

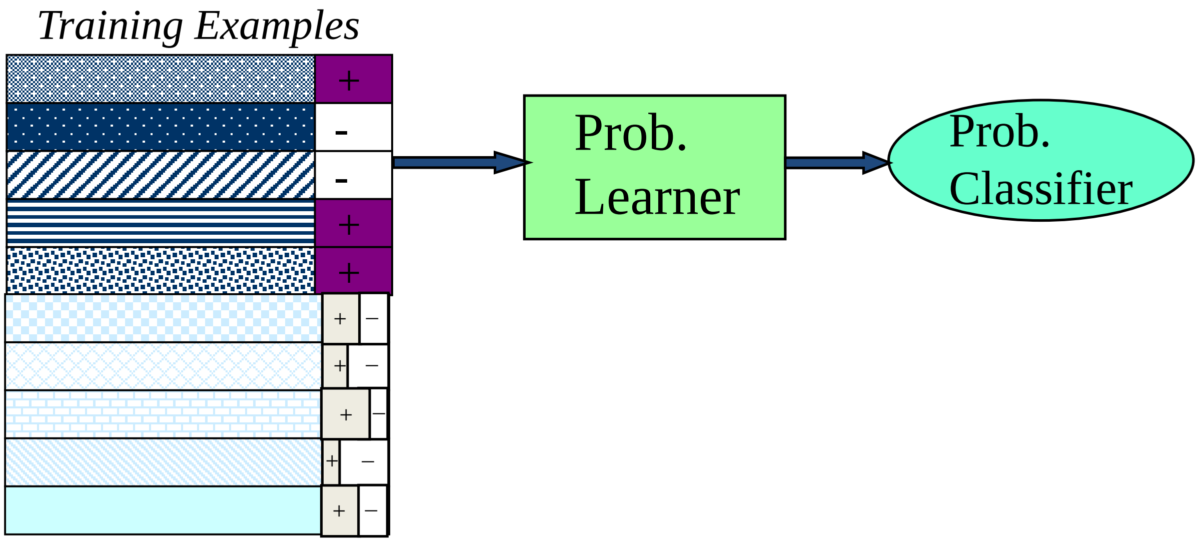


Training Examples

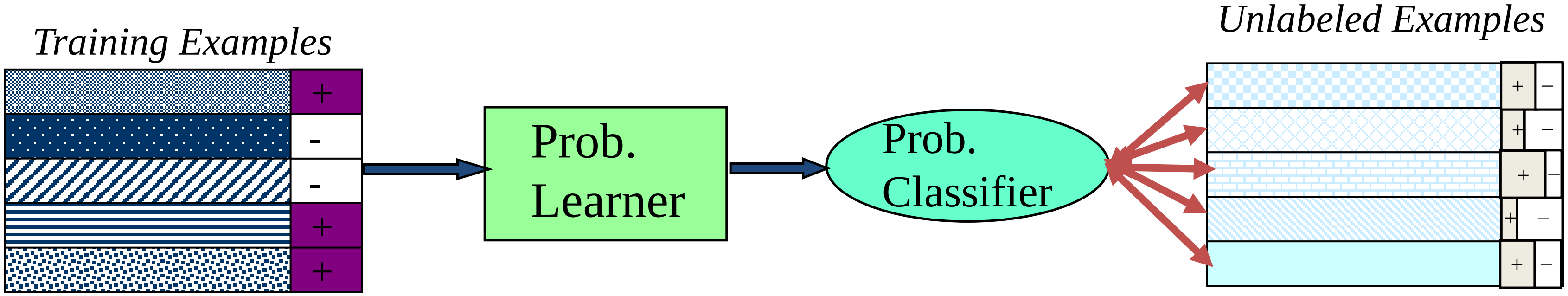
Prob. Learner

Prob. Classifier

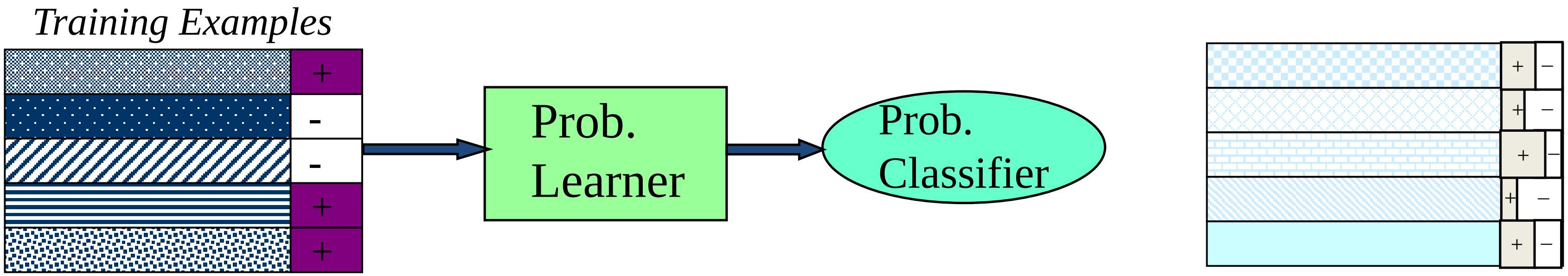
Semi-Supervised EM



Semi-Supervised EM



Semi-Supervised EM



Continue retraining iterations until probabilistic labels on unlabeled data converge.

Semi-Supervised Results

- Use of additional unlabeled data improves on supervised learning when amount of labeled data is very small and amount of unlabeled data is large.
- Can degrade performance when there is sufficient labeled data to learn a decent model and when unsupervised learning tends to create labels that are incompatible with the desired ones.
  - There are negative results for semi-supervised POS tagging since unsupervised learning tends to learn semantic labels (e.g. eating verbs, animate nouns) that are better at predicting the data than purely syntactic labels (e.g. verb, noun).

Conclusions

- POS Tagging is the lowest level of syntactic analysis.
- It is an instance of sequence labeling, that also has applications in information extraction, phrase chunking, semantic role labeling, and bioinformatics.
- HMMs are a standard generative probabilistic model for sequence labeling that allows for efficiently computing the globally most probable sequence of labels and supports supervised, unsupervised and semi-supervised learning.