Hidden Markov Models (HMMs)

Raymond J. Mooney

University of Texas at Austin

1

Part Of Speech Tagging

- Annotate each word in a sentence with a part-of-speech marker.
- Lowest level of syntactic analysis.

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

- Useful for subsequent syntactic parsing and word sense disambiguation.

2

English POS Tagsets

- Original Brown corpus used a large set of 87 POS tags.
- Most common in NLP today is the Penn Treebank set of 45 tags.
  - Tagset used in these slides.
  - Reduced from the Brown set for use in the context of a parsed corpus (i.e. treebank).
- The C5 tagset used for the British National Corpus (BNC) has 61 tags.

3

English Parts of Speech

- Noun (person, place or thing)
  - Singular (NN): dog, fork
  - Plural (NNS): dogs, forks
  - Proper (NNP, NNPS): John, Springfields
  - Personal pronoun (PRP): I, you, he, she, it
  - Wh-pronoun (WP): who, what
- Verb (actions and processes)
  - Base, infinitive (VB): eat
  - Past tense (VBD): ate
  - Gerund (VBG): eating
  - Past participle (VBN): eaten
  - Non 3rd person singular present tense (VBP): eat
  - 3rd person singular present tense: (VBZ): eats
  - Modal (MD): should, can
  - To (TO): to (to eat)

4

English Parts of Speech (cont.)

- Adjective (modify nouns)
  - Basic (JJ): red, tall
  - Comparative (JJR): redder, taller
  - Superlative (JJS): reddest, tallest
- Adverb (modify verbs)
  - Basic (RB): quickly
  - Comparative (RBR): quicker
  - Superlative (RBS): quickest
- Preposition (IN): on, in, by, to, with
- Determiner:
  - Basic (DT) a, an, the
  - WH-determiner (WDT): which, that
- Coordinating Conjunction (CC): and, but, or,
- Particle (RP): off (took off), up (put up)

5

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)

6

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.

7

POS Tagging Process

- Usually assume a separate initial tokenization process that separates and/or disambiguates punctuation, including detecting sentence boundaries.
- Degree of ambiguity in English (based on Brown corpus)
  - 11.5% of word types are ambiguous.
  - 40% of word tokens are ambiguous.
- Average POS tagging disagreement amongst expert human judges for the Penn treebank was 3.5%
  - Based on correcting the output of an initial automated tagger, which was deemed to be more accurate than tagging from scratch.
- Baseline: Picking the most frequent tag for each specific word type gives about 90% accuracy
  - 93.7% if use model for unknown words for Penn Treebank tagset.

8

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)
- Generally, learning-based approaches have been found to be more effective overall, taking into account the total amount of human expertise and effort involved.

Classification Learning

- Typical machine learning addresses the problem of classifying a feature-vector description into a fixed number of classes.
- There are many standard learning methods for this task:
  - Decision Trees and Rule Learning
  - Naïve Bayes and Bayesian Networks
  - Logistic Regression / Maximum Entropy (MaxEnt)
  - Perceptron and Neural Networks
  - Support Vector Machines (SVMs)
  - Nearest-Neighbor / Instance-Based

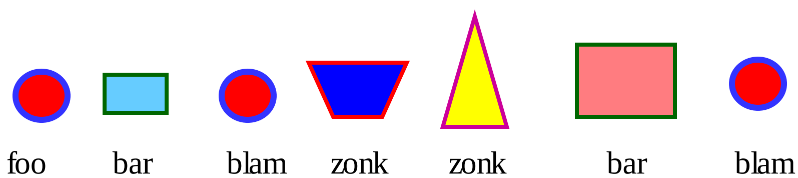
Beyond Classification Learning

- Standard classification problem assumes individual cases are disconnected and independent (i.i.d.: independently and identically distributed).
- Many NLP problems do not satisfy this assumption and involve making many connected decisions, each resolving a different ambiguity, but which are mutually dependent.
- More sophisticated learning and inference techniques are needed to handle such situations in general.

11

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).



12

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.

13

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"

14

Bioinformatics

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

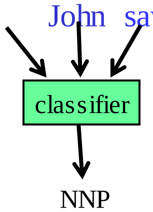
  extron intron

  – AGCTAACGTTCGATACGGATTACAGCCT

Sequence Labeling as Classification

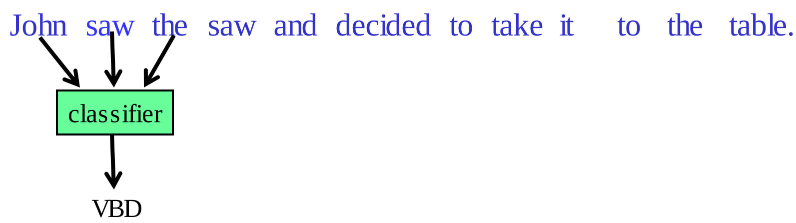
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

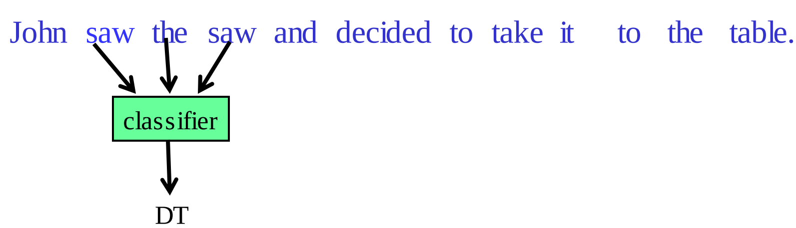
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



17

Sequence Labeling as Classification

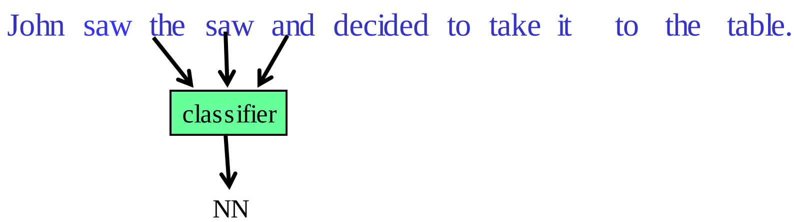
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



18

Sequence Labeling as Classification

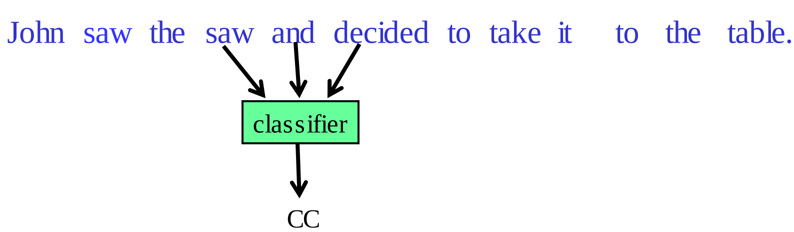
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



19

Sequence Labeling as Classification

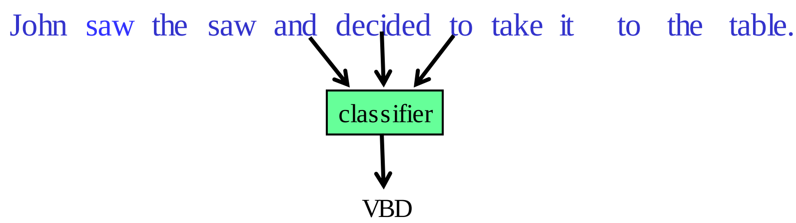
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



20

Sequence Labeling as Classification

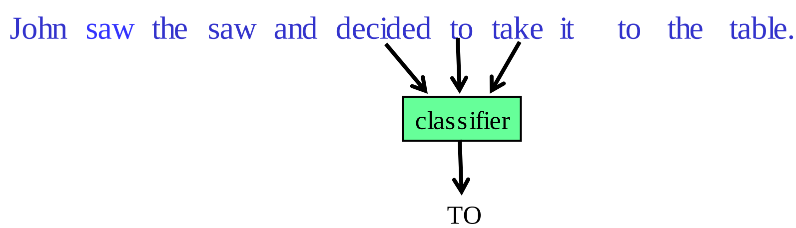
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



21

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

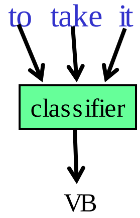


22

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

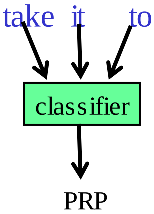
John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

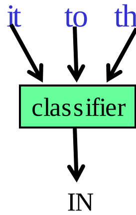
John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.

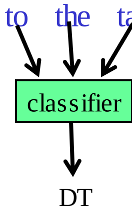


25

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

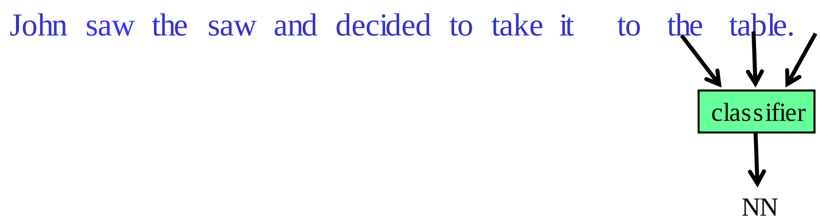
John saw the saw and decided to take it to the table.



26

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



27

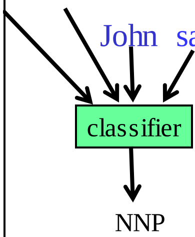
Sequence Labeling as Classification Using Outputs as Inputs

- Better input features are usually the categories of the surrounding tokens, but these are not available yet.
- Can use category of either the preceding or succeeding tokens by going forward or back and using previous output.

28

Forward Classification

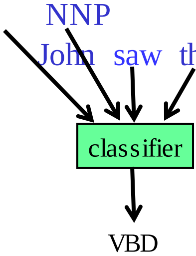
John saw the saw and decided to take it to the table.



29

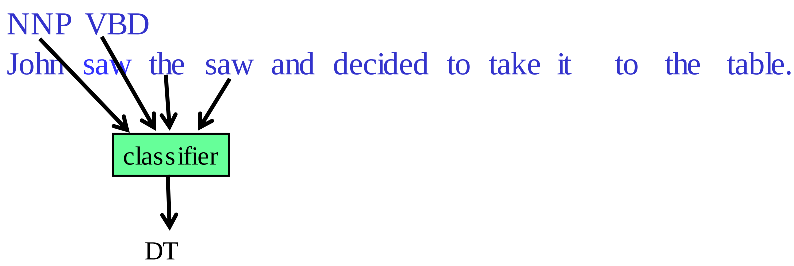
Forward Classification

NNP
John saw the saw and decided to take it to the table.



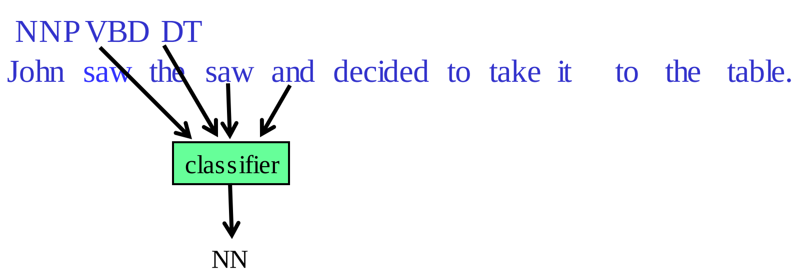
30

Forward Classification



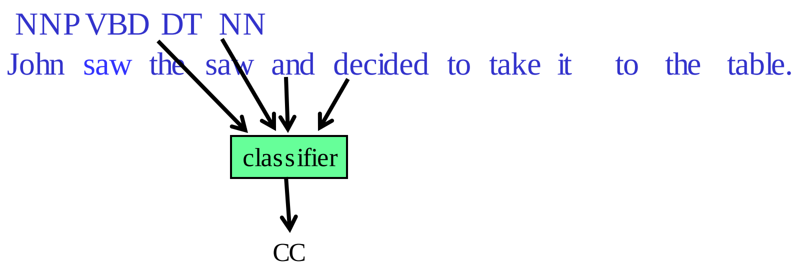
31

Forward Classification

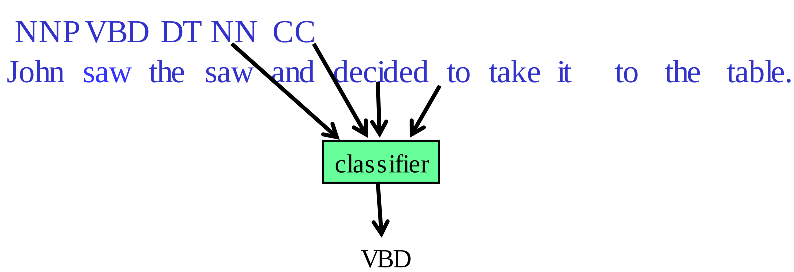


32

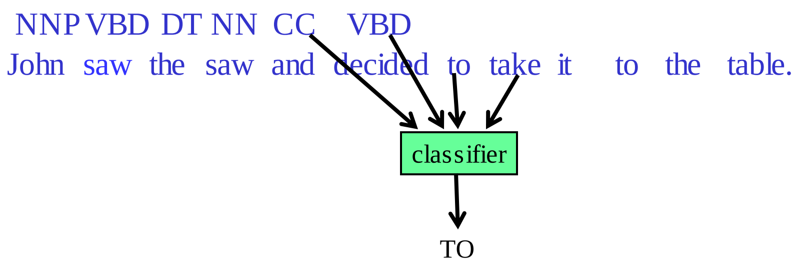
Forward Classification



Forward Classification

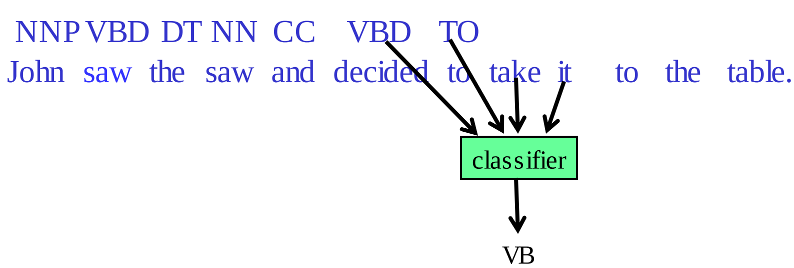


Forward Classification



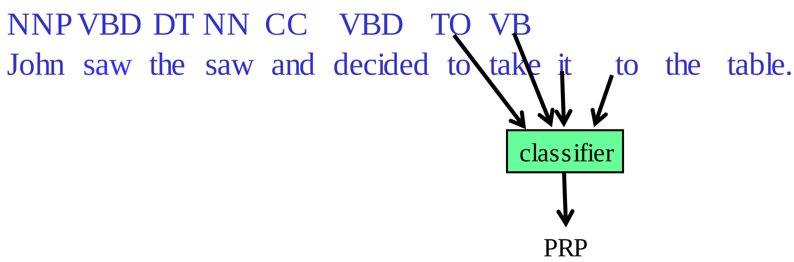
35

Forward Classification



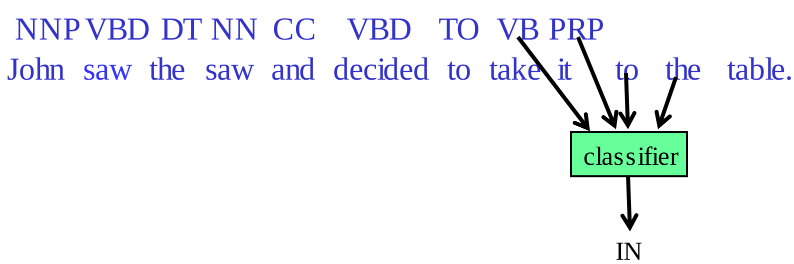
36

Forward Classification



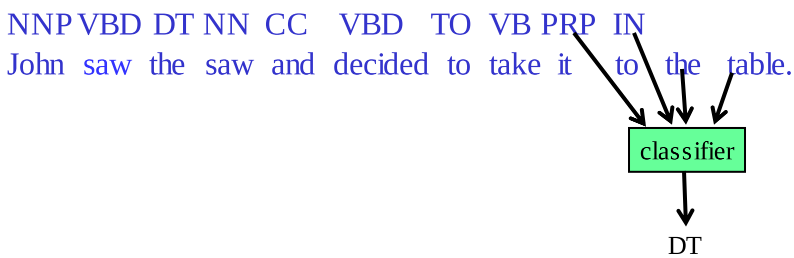
37

Forward Classification

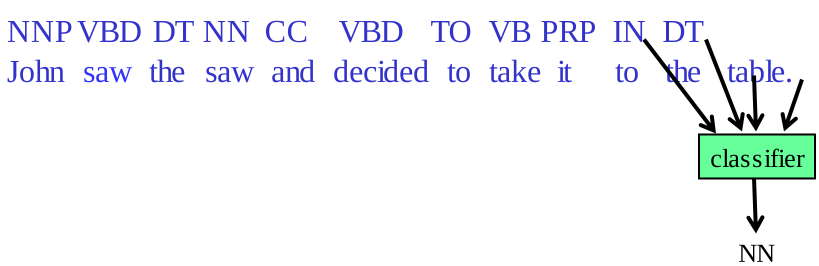


38

Forward Classification

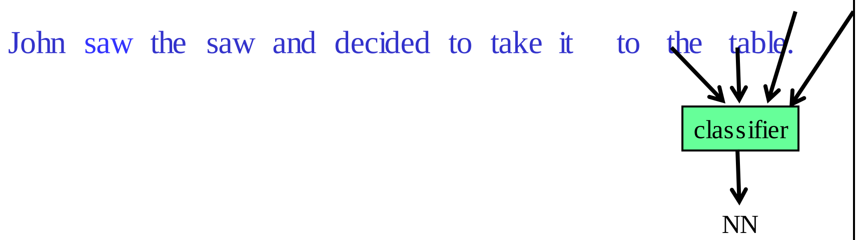


Forward Classification



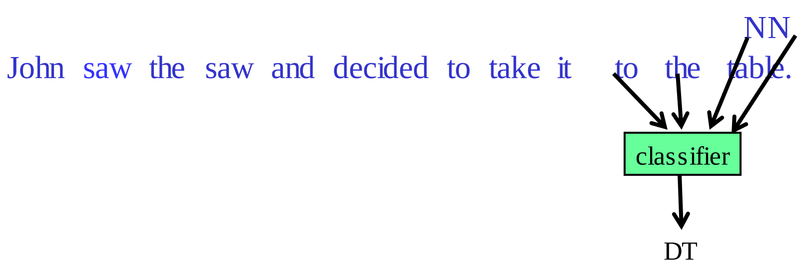
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



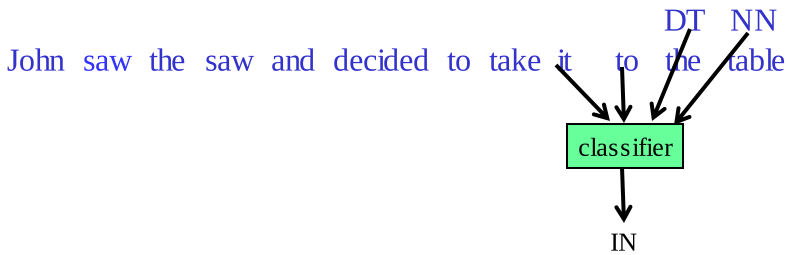
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



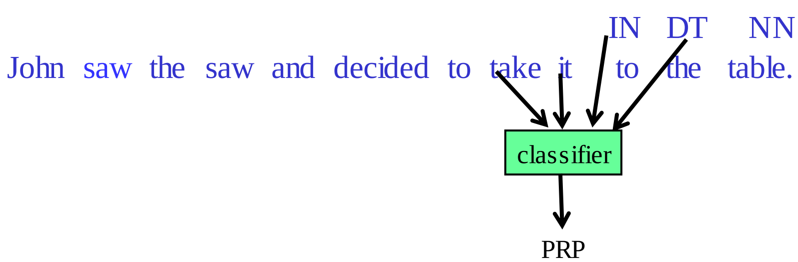
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



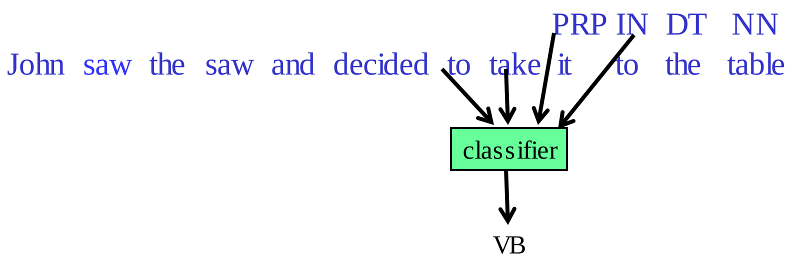
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



Backward Classification

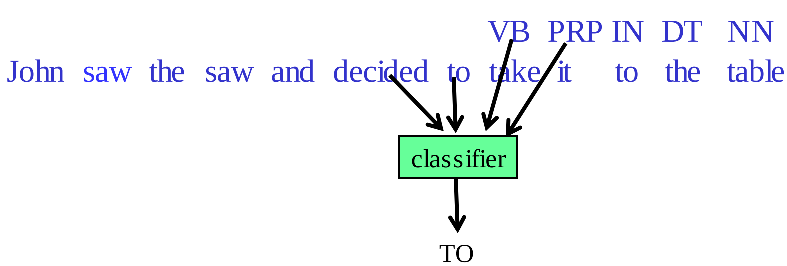
- Disambiguating “to” in this case would be even easier backward.



45

Backward Classification

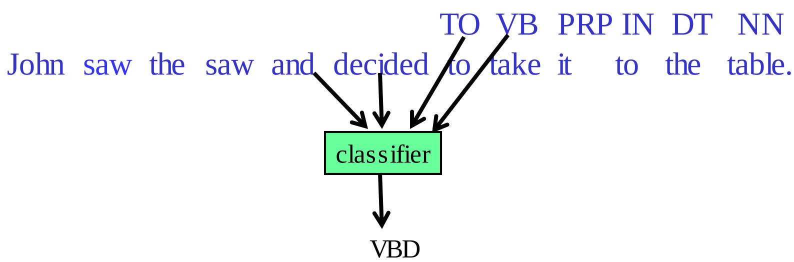
- Disambiguating “to” in this case would be even easier backward.



46

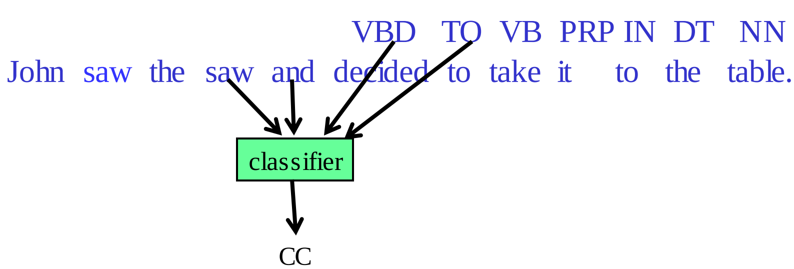
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



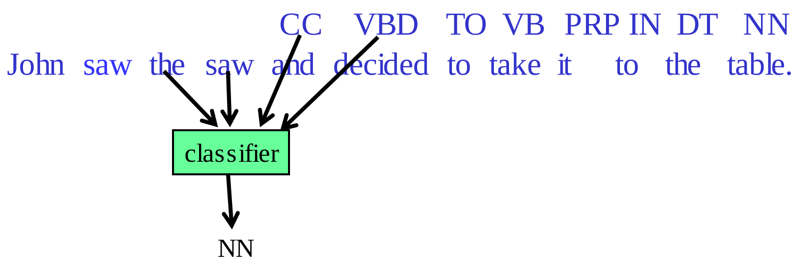
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



Backward Classification

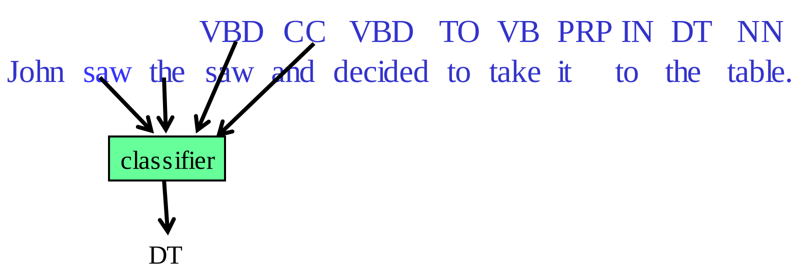
- Disambiguating “to” in this case would be even easier backward.



49

Backward Classification

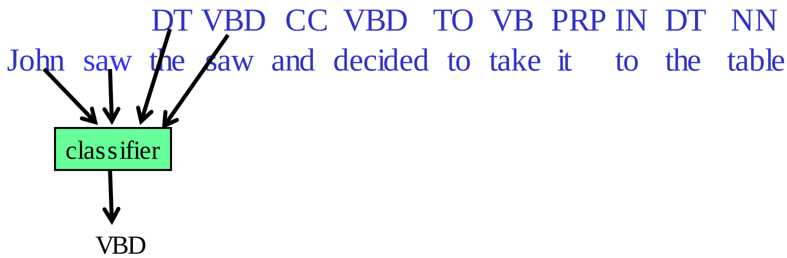
- Disambiguating “to” in this case would be even easier backward.



50

Backward Classification

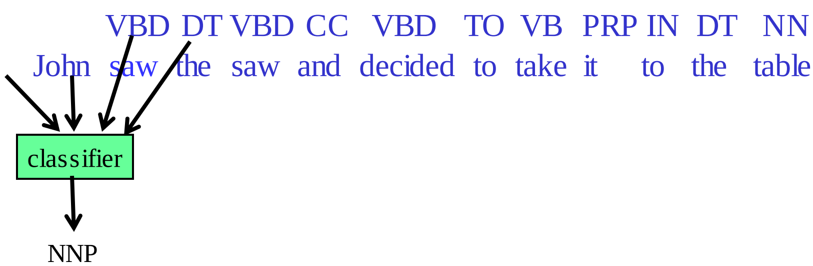
- Disambiguating “to” in this case would be even easier backward.



51

Backward Classification

- Disambiguating “to” in this case would be even easier backward.



52

Problems with Sequence Labeling as Classification

- Not easy 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.

53

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)

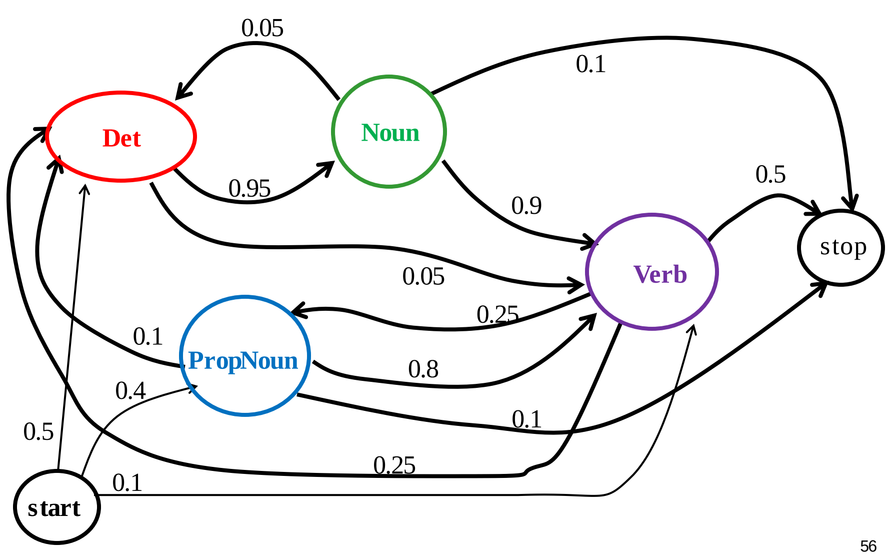
54

Markov Model / Markov Chain

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

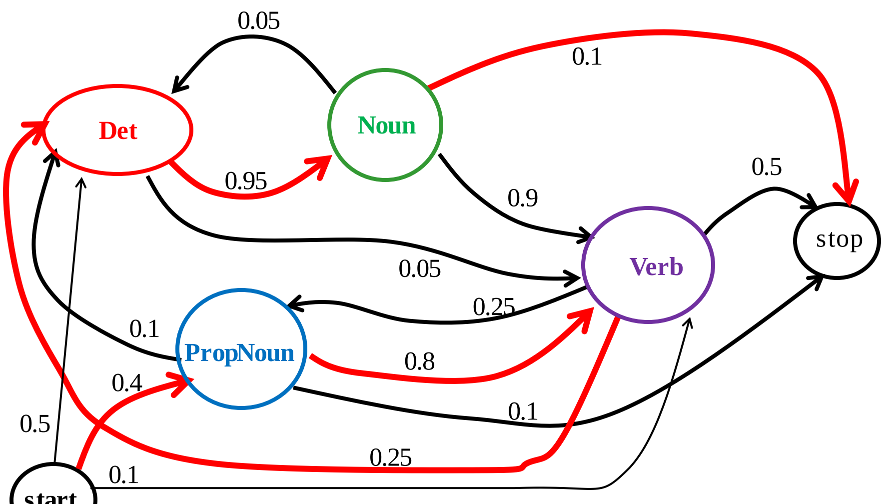
55

Sample Markov Model for POS



56

Sample Markov Model for POS



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

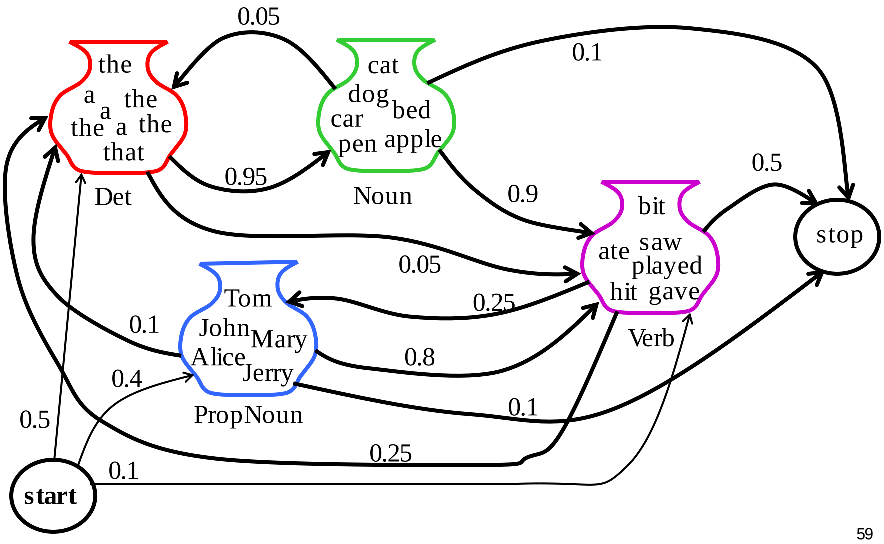
57

Hidden Markov Model

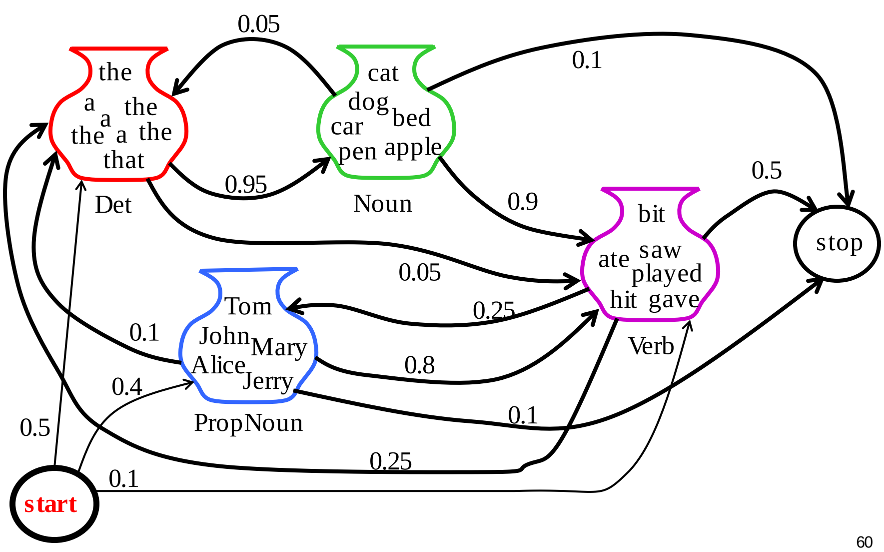
- Probabilistic generative model for sequences.
- Assume an underlying set of ***hidden*** (unobserved) states in which the model can be (e.g. parts of speech).
- Assume probabilistic transitions between states over time (e.g. transition from POS to another POS as sequence is generated).
- Assume a ***probabilistic*** generation of tokens from states (e.g. words generated for each POS).

58

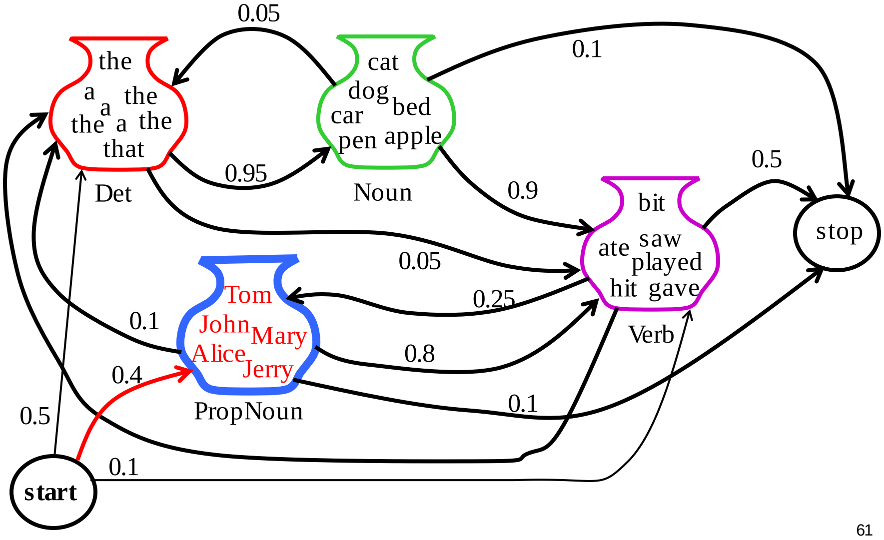
Sample HMM for POS



Sample HMM Generation

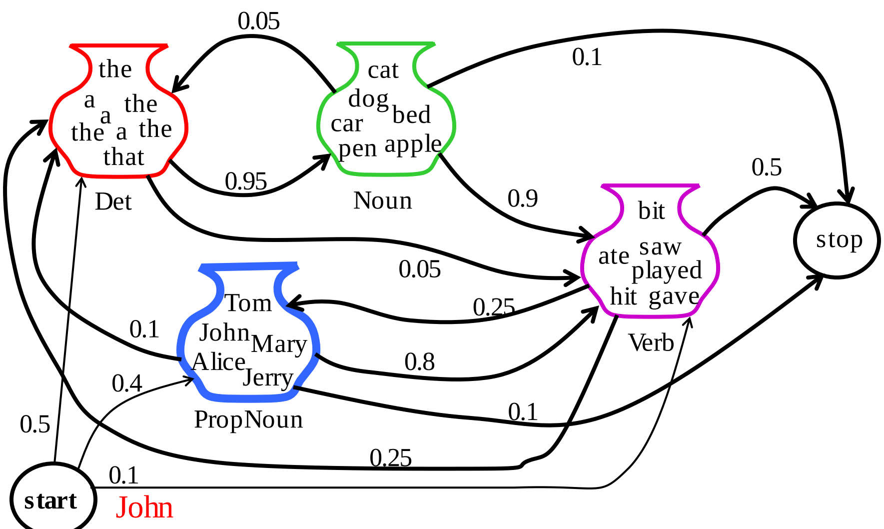


Sample HMM Generation



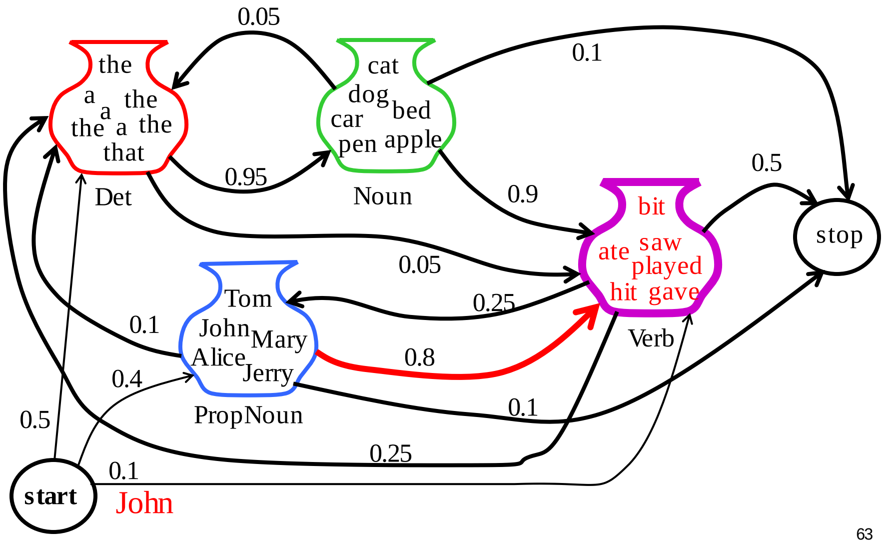
61

Sample HMM Generation



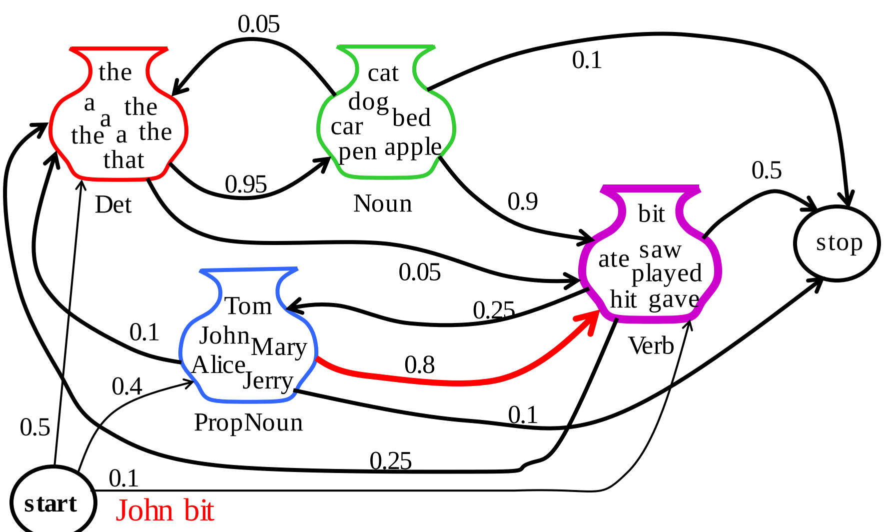
62

Sample HMM Generation



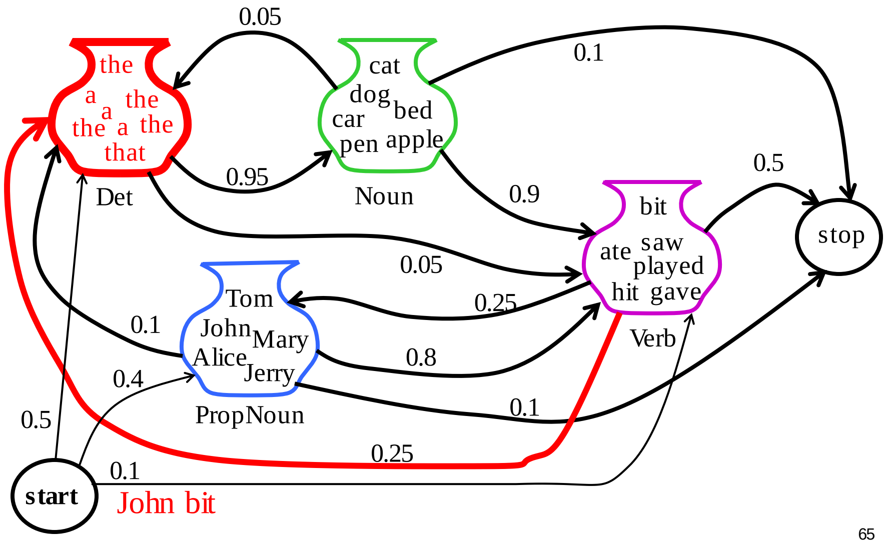
63

Sample HMM Generation



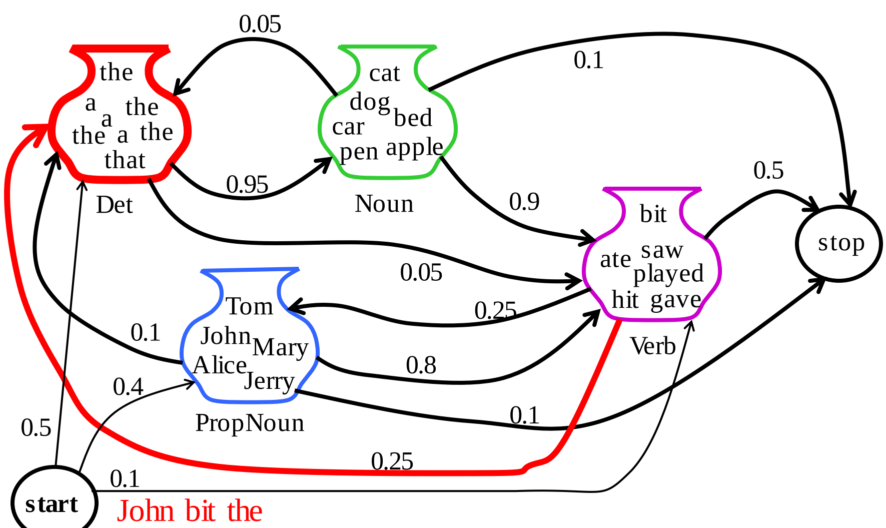
64

Sample HMM Generation



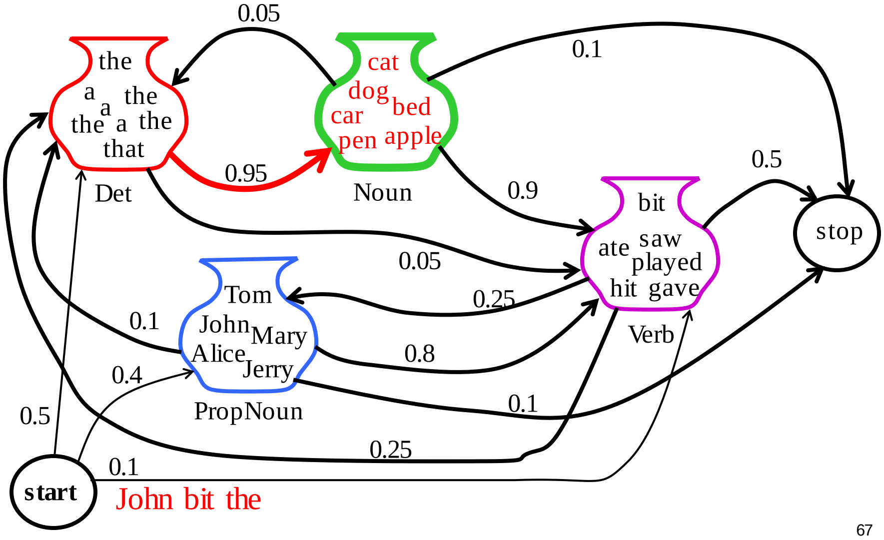
65

Sample HMM Generation



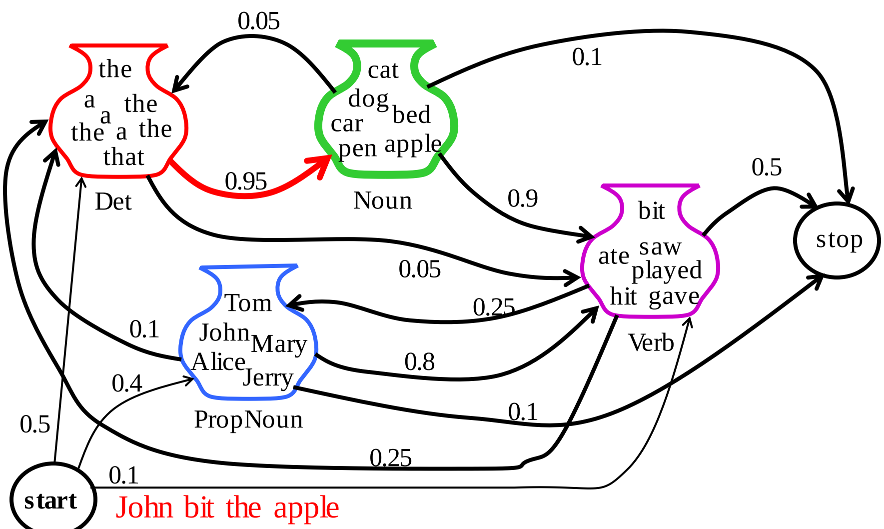
66

Sample HMM Generation



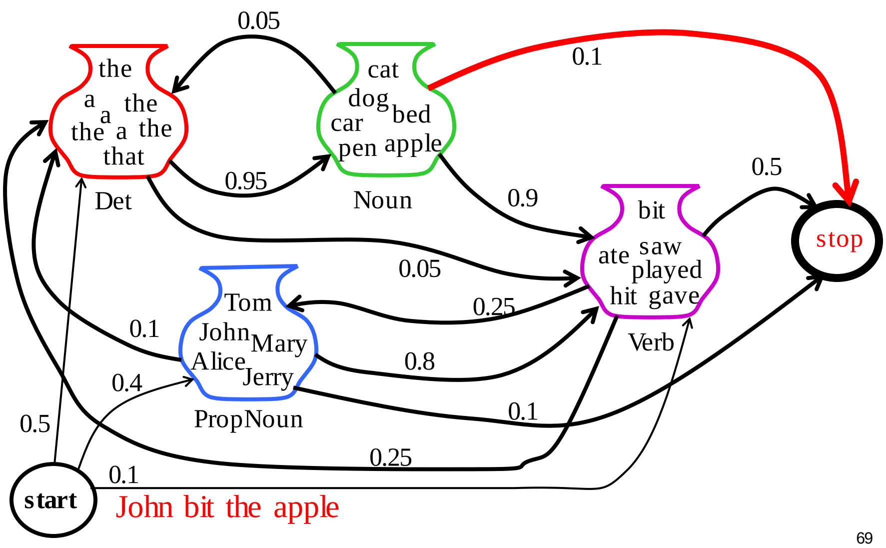
67

Sample HMM Generation



68

Sample HMM Generation



69

Formal Definition of an HMM

- A set of $N+2$ states $S=\{s_0,s_1,s_2, \ldots s_N, s_F\}$
  - Distinguished start state: s_0
  - Distinguished final state: 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 $\lambda=\{A,B\}$

70

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 $b_{qt}(k)$

71

Three Useful HMM Tasks

- Observation Likelihood: To classify and order sequences.
- Most likely state sequence (Decoding): To tag each token in a sequence with a label.
- Maximum likelihood training (Learning): To train models to fit empirical training data.

72

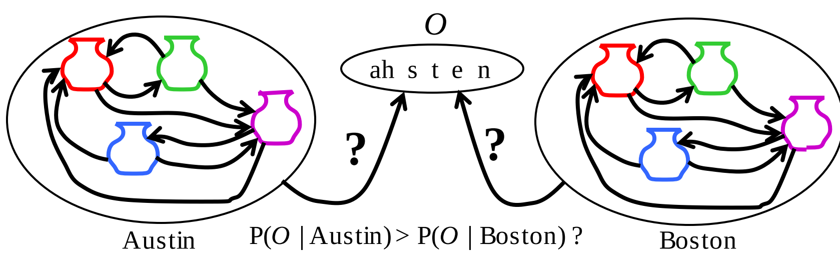
HMM: Observation Likelihood

- Given a sequence of observations, *O*, and a model with a set of parameters, λ, what is the probability that this observation was generated by this model: P(O| λ) ?
- Allows HMM to be used as a language model: A formal probabilistic model of a language that assigns a probability to each string saying how likely that string was to have been generated by the language.
- Useful for two tasks:
  - Sequence Classification
  - Most Likely Sequence

73

Sequence Classification

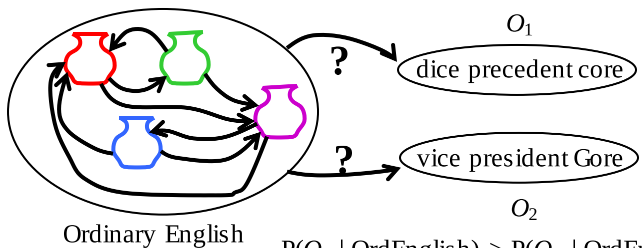
- Assume an HMM is available for each category (i.e. language).
- What is the most likely category for a given observation sequence, i.e. which category's HMM is most likely to have generated it?
- Used in speech recognition to find most likely word model to have generate a given sound or phoneme sequence.



74

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.



$P(O_2 \mid \text{OrdEnglish}) > P(O_1 \mid \text{OrdEnglish})$?

75

HMM: Observation Likelihood Naïve Solution

- Consider all possible state sequences, Q, of length T that the model could have traversed in generating the given observation sequence.
- Compute the probability of a given state sequence from A, and multiply it by the probabilities of generating each of given observations in each of the corresponding states in this sequence to get $P(O,Q \mid \lambda) = P(O \mid Q, \lambda)\, P(Q \mid \lambda)$.
- Sum this over all possible state sequences to get $P(O \mid \lambda)$.
- Computationally complex: $O(TN^T)$.

76

HMM: Observation Likelihood Efficient Solution

- Due to the Markov assumption, the probability of being in any state at any given time *t* only relies on the probability of being in each of the possible states at time $t-1$.
- Forward Algorithm: Uses dynamic programming to exploit this fact to efficiently compute observation likelihood in $O(TN^2)$ time.
  - Compute a ***forward trellis*** that compactly and implicitly encodes information about all possible state paths.

77

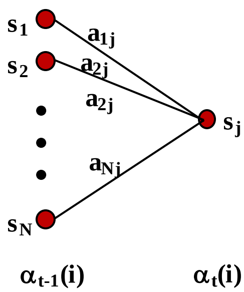
Forward Probabilities

- 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*).

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

78

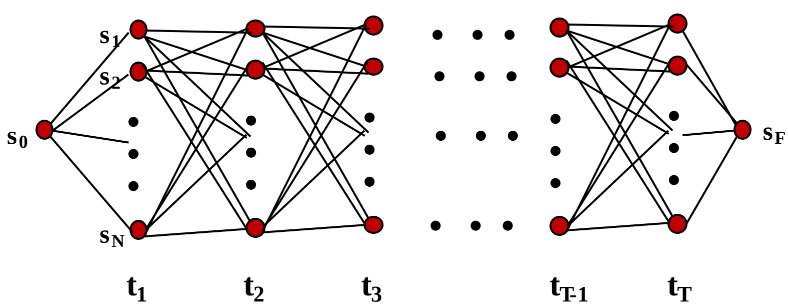
Forward Step



- 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.

79

Forward Trellis



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

80

Computing the Forward Probabilities

- Initialization

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

- Recursion

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

- Termination

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

81

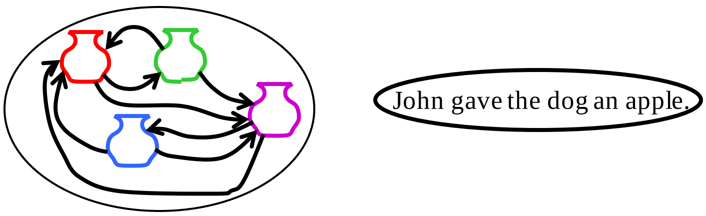
Forward Computational Complexity

- Requires only O(TN^2) time to compute the probability of an observed sequence given a model.
- Exploits the fact that all state sequences must merge into one of the N possible states at any point in time and the Markov assumption that only the last state effects the next one.

82

Most Likely State Sequence (Decoding)

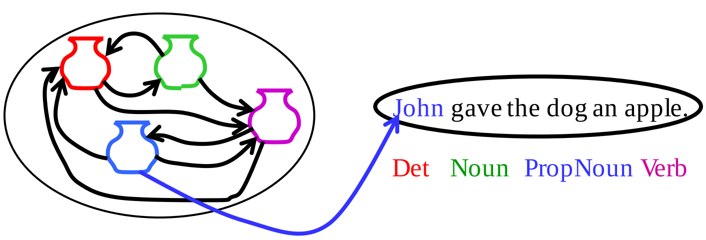
- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.



83

Most Likely State Sequence

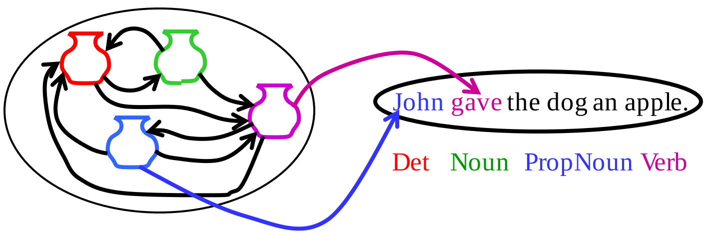
- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.



84

Most Likely State Sequence

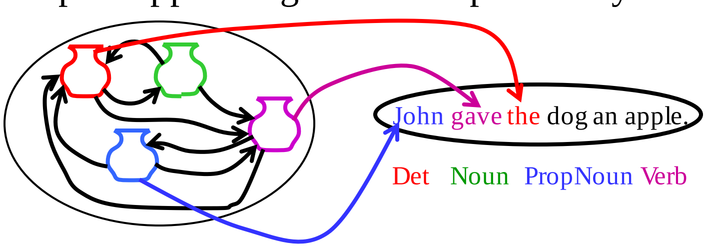
- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.



85

Most Likely State Sequence

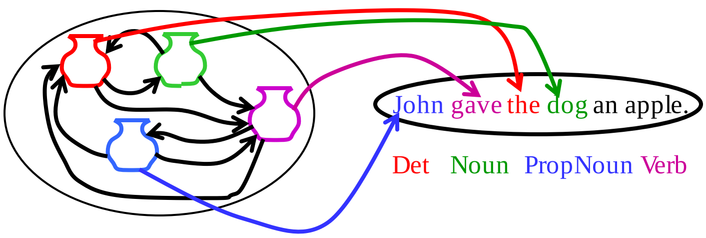
- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.



86

Most Likely State Sequence

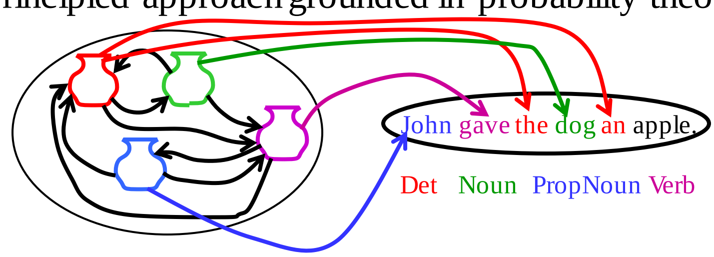
- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.



87

Most Likely State Sequence

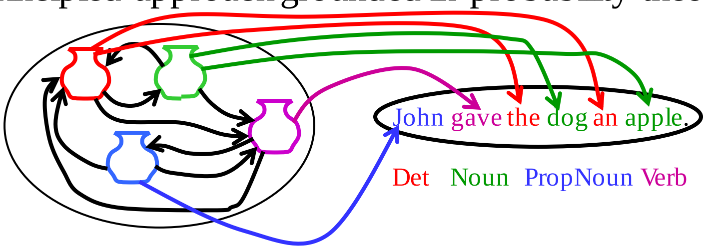
- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.



88

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?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.



89

HMM: Most Likely State Sequence Efficient Solution

- Obviously, could use naïve algorithm based on examining every possible state sequence of length T.
- Dynamic Programming can also be used to exploit the Markov assumption and efficiently determine the most likely state sequence for a given observation and model.
- Standard procedure is called the Viterbi algorithm (Viterbi, 1967) and also has $O(N^2T)$ time complexity.

90

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).

91

Computing the Viterbi Scores

- 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

92

Computing the Viterbi Backpointers

- Initialization

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

- Recursion

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

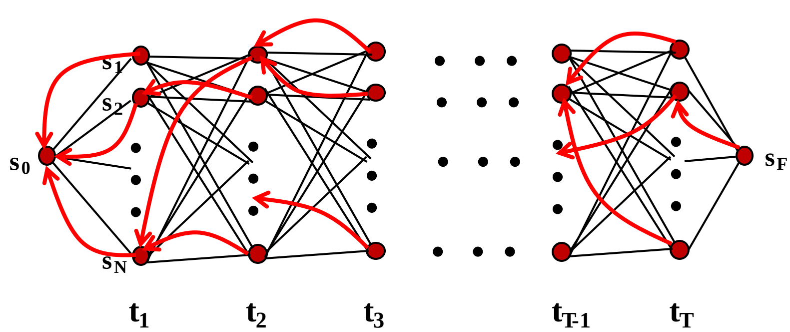
- Termination

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

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

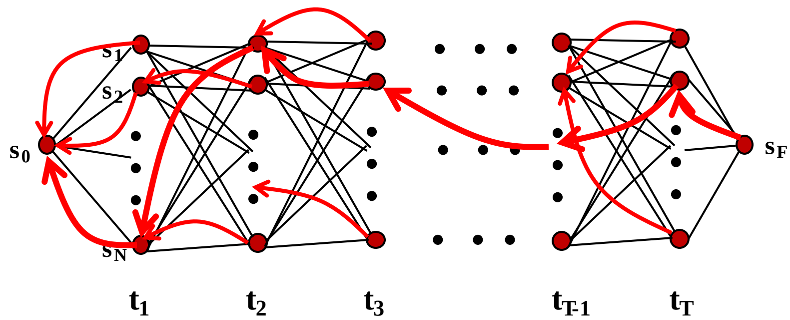
93

Viterbi Backpointers



94

Viterbi Backtrace



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

95

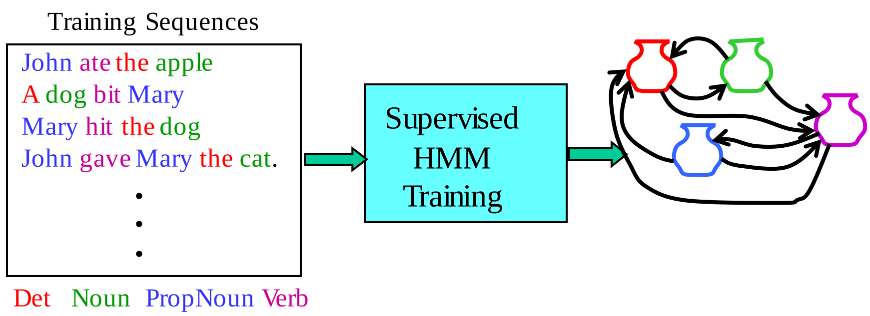
HMM Learning

- **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.

96

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.



Supervised Parameter Estimation

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

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

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

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

- Use appropriate smoothing if training data is sparse.

Learning and Using HMM Taggers

- Use a corpus of labeled sequence data to easily 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.

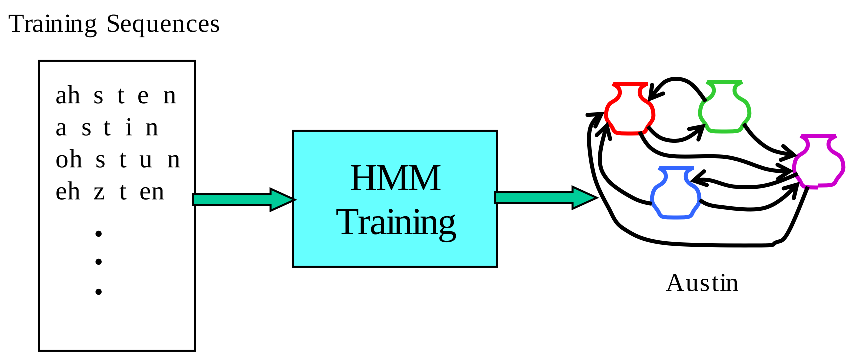
99

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.

100

Unsupervised
Maximum Likelihood Training



101

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.
- Only need to have an unannotated observation sequence (or set of sequences) generated from the model. Does not need to know the correct state sequence(s) for the observation sequence(s). In this sense, it is unsupervised.

102

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.

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

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

- 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 Lapacian 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 $P(O|\lambda)$.
- 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 $P(O|\lambda)$.
- In practice, EM is able to find a good set of parameters that provide a good fit to the training data in many cases.

105

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.

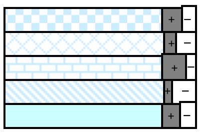
106

EM

Initialize:

Assign random probabilistic labels to unlabeled data

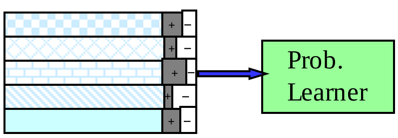
Unlabeled Examples



EM

Initialize:

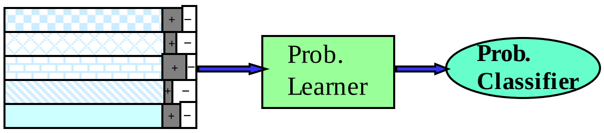
Give soft-labeled training data to a probabilistic learner



EM

Initialize:

Produce a probabilistic classifier

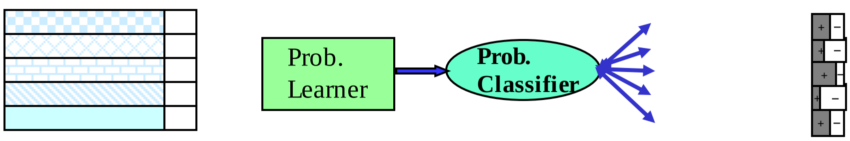


109

EM

E Step:

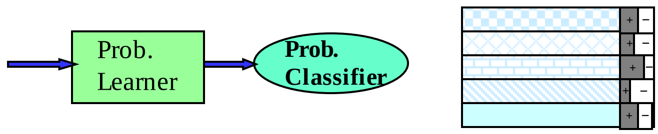
Relabel unlabled data using the trained classifier



110

EM

M step:
Retrain classifier on relabeled data



Continue EM iterations until probabilistic labels on unlabeled data converge.

111

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

Assume an HMM with N states.
Randomly set its parameters $\lambda=(A,B)$
(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

112

Backward Probabilities

- Let $\beta_t(i)$ be the probability of observing the final set of observations from time t+1 to T given that one is in state i at time t.

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots o_T \mid q_t = s_{i,} \lambda)$$

113

Computing the Backward Probabilities

- Initialization

$$\beta_T(i) = a_{iF} \quad 1 \le i \le N$$

- Recursion

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad 1 \le i \le N, \; 1 \le t < T$$

- Termination

$$P(O \mid \lambda) = \alpha_T(s_F) = \beta_1(s_0) = \sum_{j=1}^{N} a_{0j} b_j(o_1) \beta_1(j)$$

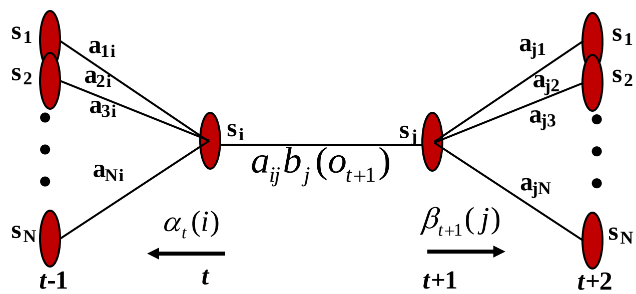
114

Estimating Probability of State Transitions

- Let $\xi_t(i,j)$ be the probability of being in state i at time t and state j at time $t+1$

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

$$\xi_t(i,j) = \frac{P(q_t = s_i, q_{t+1} = s_j, O \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)}$$



Re-estimating A

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to } j}{\text{expected number of transitions from state } i}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)}$$

Estimating Observation Probabilities

- Let $\gamma_t(i)$ be the probability of being in state i at time t given the observations and the model.

$$\gamma_t(j) = P(q_t = s_j \mid O, \lambda) = \frac{P(q_t = s_j, O \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(j)\beta_t(j)}{P(O \mid \lambda)}$$

Re-estimating B

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

$$\hat{b}_j(v_k) = \frac{\sum_{\mathrm{t=1,\,s.t.\,o_t}=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Pseudocode for Baum-Welch (EM) Algorithm for Training HMMs

Assume an HMM with N states.
Randomly set its parameters $\lambda=(A,B)$
(making sure they represent legal distributions)
Until converge (i.e. λ no longer changes) do:
E Step:
Compute values for $\gamma_t(j)$ and $\xi_t(i,j)$ using current values for parameters A and B.
M Step:
Re-estimate parameters:

$$a_{ij} = \hat{a}_{ij}$$
$$b_j(v_k) = \hat{b}_j(v_k)$$

119

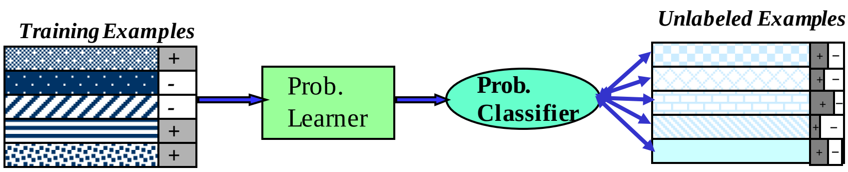
EM Properties

- 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

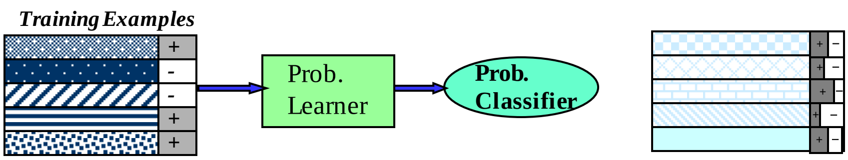
- EM algorithms can be trained with a mix of labeled and unlabeled data.
- EM basically predicts a probabilistic (soft) labeling of the instances and then iteratively retrains using supervised learning on these predicted labels ("self training").
- EM can also exploit supervised data:
  - 1) Use supervised 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



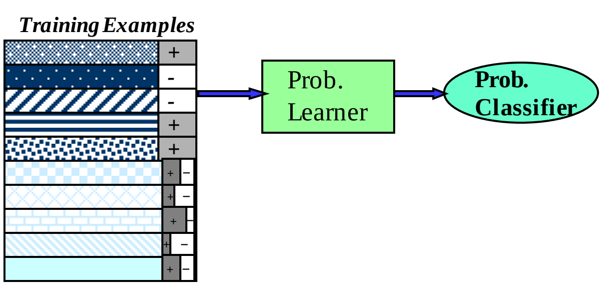
122

Semi-Supervised EM



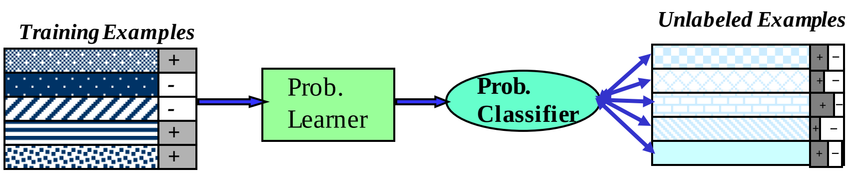
123

Semi-Supervised EM

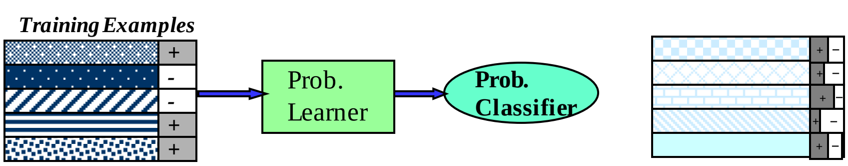


124

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, a collective classification task 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.