Natural Language Processing: Part-Of-Speech Tagging, Sequence Labeling, dan Hidden Markov Models (HMMs)

Ahmad Rio Adriansyah, M.Si.

STT Terpadu Nurul Fikri

*diadaptasi dari slide Raymond J. Mooney (Univ. Texas)

Part Of Speech Tagging

- Menganotasi setiap kata pada kalimat dengan tanda “part-of-speech”
- Level terendah dari analisis sintaksis.

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

- Berguna untuk **syntactic parsing** dan **word sense disambiguation**.

English POS Tagsets

- Original Brown corpus menggunakan 87 buah POS tag.
- Yang paling umum digunakan dalam NLP dewasa ini adalah Penn Treebank yang terdiri dari 45 buah POS tag.
  - Tagset digunakan pada slide ini (bahasa inggris).
  - Disederhanakan dari Brown set untuk digunakan pada konteks parsed corpus (i.e. treebank).
- Tagset C5 yang digunakan British National Corpus (BNC) memiliki buah 61 tag.

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)

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)

POS Tagset Bahasa Indonesia

- Adriani (2009)
  - 37 tag untuk kata bahasa Indonesia, termasuk tanda baca
  - Disederhanakan menjadi 25 tag dengan menggabungkan beberapa kategori menjadi satu
- MorphInd Tagset (Larasati, 2011)
  - 3 tingkatan tag
  - 19 tag di posisi pertama
- Dinakaramani (2014)
  - 23 tag hasil penyempurnaan tagset sebelumnya

POS Tagset Bahasa Indonesia

- INACL (2017)
  - 26 tag mewakili kelas kata dalam 12 kategori
  - 10 tag untuk label kategori frasa
  - 5 tag untuk label penggabung kata (+NNO, +NNP, +PPO, +CCN, +CSN)
  - Hasil konvensi (kesepakatan) asosiasi komputasi linguistik di indonesia (INACL)

POS Tag Bahasa Indonesia

- Penulisan tag dalam korpus biasanya mengikuti tokennya dan dipisahkan oleh garis miring (/)
- Contoh :

  “Pengetahuan/NN bahasa/NN dan/CC ilmu komputer/NNP sangat/RB penting/JJ dalam/IN mempelajari/VB NLP/NNP.”

Closed vs. Open Class

- Kategori ***Closed class*** terdiri dari sekumpulan kecil kata yang tidak berubah (fixed) yang berfungsi secara gramatikal pada bahasa tertentu.
  - (Eng) Pronouns, Prepositions, Modals, Determiners, Particles, Conjunctions
  - (Ind) Kata hubung, kata depan
- Kategori ***Open class*** memiliki banyak kata dan kata-kata baru dapat dengan mudah diciptakan.
  - (Eng) Nouns (Googler, textlish), Verbs (Google), Adjectives (geeky), Abverb (automagically)
  - (Ind) Kata benda (tongsis, selfie), Kata sifat (jayus), Kata kerja (ngegoogle)

Ambiguity in POS Tagging (bahasa Inggris)

- "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.

Ambiguity in POS Tagging (tagset Dinakaramani)

- “Bisa” dapat berupa kata benda (NN) atau modal (MD)
  - Saya bisa/MD mengerjakan soal NLP.
  - Ular itu mengeluarkan bisa/NN dari taringnya.
- “Apel” dapat berupa kata kerja (VB) atau kata benda (NN)
  - Cecep makan apel/NN tanpa dikupas terlebih dahulu.
  - Para satpam melakukan apel/VB pagi di lapangan.

Ambiguity in POS Tagging (tagset INACL)

- Bandingkan kata “lagi”:
  - Dia datang **lagi** ke rumahku (ADV)
  - Dia **lagi** bersiap-siap ke sekolah (ADK)
  - Dia cantik **lagi** pandai (CCN)
- Bandingkan kata “sama”:
  - Kemampuan setiap anak tidak **sama** (ADJ)
  - Mereka tidak suka **sama** sikapnya (PPO)

Teks Preprocessing

Untuk mempermudah komputasi, biasanya dilakukan hal hal berikut terhadap kalimat sebelum diolah lebih lanjut (proses ini mempermudah komputasi tetapi menghilangkan beberapa informasi dan mengurangi akurasi) :

- **Case folding** = mengubah semua huruf menjadi huruf kecil. karakter selain huruf a-z dihilangkan dan dianggap delimiter
- **Tokenizing** = memotong string input berdasarkan kata-kata yang menyusunnya
- **Stopword removal (filtering)** = mengambil kata kata penting dari token. dapat menggunakan stoplist (membuang kata yang tidak deskriptif) atau wordlist (menyimpan kata yang penting)
- **Stemming** = mencari akar kata (root word) dari hasil filtering

Ronen Feldman and James Sanger. *The Text Mining Handbook*. 2007.

POS Tagging Process

- Proses tokenisasi (tokenization) : memisahkan dan/atau mendisambiguasikan tanda baca, termasuk mendeteksi batas kalimat (sentence boundaries).
- Derajat ambiguitas bahasa Inggris (bds. Brown corpus)
  - 11.5% dari tipe kata (word types) adalah ambigu.
  - 40% dari token (word tokens) adalah ambigu.
- Rata-rata perbedaan pendapat untuk POS tagging (Average POS tagging disagreement) diantara para ahli pada Penn treebank sebesar 3.5%
  - Berdasarkan output dari automated tagger yang dikoreksi. Automated tagger dianggap lebih akurat dari tagging manual dari awal.
- Baseline: Memilih tag yang paling sering muncul untuk tiap kata memberikan akurasi sekitar 90%
  - 93.7% jika menggunakan model untuk unknown words pada Penn Treebank tagset.

POS Tagging Approaches

- **Rule-Based**: Aturan yang disusun berdasarkan pengetahuan tentang leksikal atau pengetahuan bahasa lainnya.
- **Learning-Based**: Ditraining menggunakan corpora yang sudah dianotasi, contohnya 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)
- Secara umum, pendekatan learning-based lebih efektif, dengan mempertimbangkan banyak ahli dan usaha yang terlibat.

Latihan

Beri label kalimat-kalimat berikut sesuai dengan tagset INACL (2017).

•

Sequence Labeling dan HMM

Classification Machine Learning

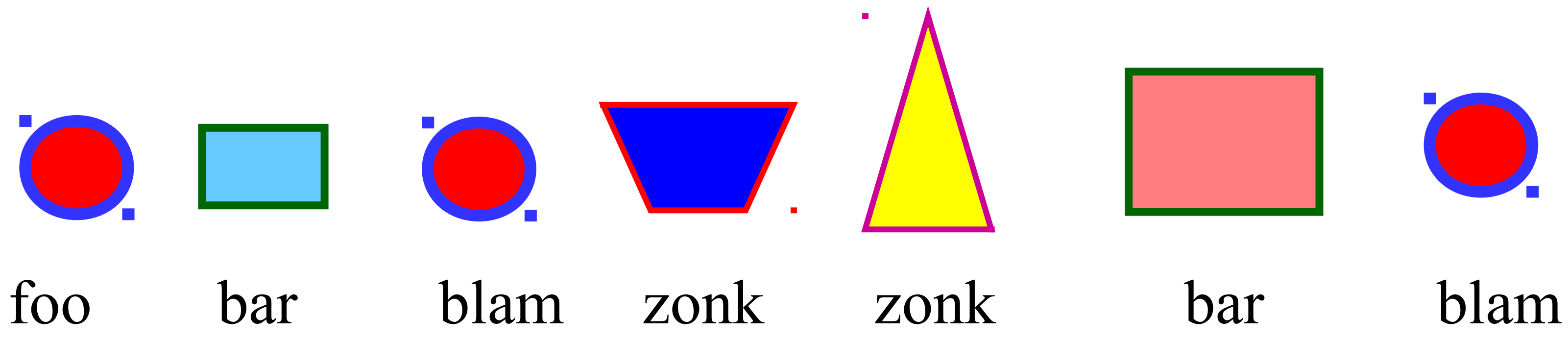
- Machine learning dapat berfungsi untuk klasifikasi ataupun regresi
- Classification machine learning memetakan masalah klasifikasi dari feature-vector ke sejumlah kelas.
- Banyak standard metode learning untuk pekerjaan ini :
  - Decision Trees and Rule Learning
  - Naïve Bayes and Bayesian Networks
  - Logistic Regression / Maximum Entropy (MaxEnt)
  - Perceptron and Neural Networks
  - Support Vector Machines (SVMs)
  - Nearest-Neighbor / Instance-Based

Beyond Classification Learning

- Klasifikasi mengasumsikan setiap kasus adalah tidak terkait dan independen (independently and identically distributed).
- Banyak masalah NLP tidak memenuhi asumsi ini (banyak keputusan yang berkaitan, dan mutually dependen)
- Teknik inferensi dan learning yang lebih canggih diperlukan untuk menangani situasi tersebut secara umum.

Sequence Labeling Problem

- Banyak masalah dalam NLP dapat dianggap sebagai pelabelan rangkaian (sequence labeling).
- Setiap token dalam rangkaian diberikan label.
- Label pada token dependen terhadap label token lainnya pada rangkaian tersebut, terutama tetangganya (tidak i.i.d).



Analogi Sequence Labeling

- Urns and Balls example
  - Anggap ada 2 buah guci berisi bola hitam dan putih
  - Guci yang satu memiliki lebih banyak bola hitam (90%-10%) dan sebaliknya
  - Seseorang mengambil sebuah bola dan menunjukkan bola yang terambil tanpa memberi tahu bola tersebut dari guci yang mana, lalu dikembalikan
  - Dia cenderung (90%) akan mengambil bola selanjutnya dari guci yang sama
  - Pengamat hanya melihat rangkaian bola yang ditunjukkan dan mencatatnya

L.R. Rabiner. *Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. 1989.

Analogi Sequence Labeling

- Urns and Balls question
  - Bisakah kita memprediksi dari guci mana bola diambil pada waktu tertentu?
  - Berapa peluang hasil pengamatan rangkaian bola yang ditunjukkan
  - Bisakah kita mempelajari rasio bola di tiap guci hanya dengan melihat hasil pengamatan jika kita tidak tahu rasio itu sebelumnya?

L.R. Rabiner. *Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. 1989.

Analogi Sequence Labeling

- Excercise Video Activity
  - Misalnya kita diberikan video senam (pesenamnya hanya seorang) dan tertarik untuk melabelinya antara "berdiri", "jongkok", atau "tiduran" (anggap tidak ada kegiatan selain itu)
  - Subjeknya ditracking dan diberikan kotak (bounding box) melingkupinya dalam tiap frame
  - Fitur yang kita jadikan masukan adalah tinggi dan lebar bounding box
  - Jadi kita punya observasi kontinu dan tiga buah label
- Pertanyaan
  - Diberikan tinggi dan lebar bounding box di semua frame, bisakah kita menentukan tipe kegiatan di tiap framenya?

Hakan Erdogan. *Structured Learning for Sequence Labeling*. 2010.

Sequence Labeling Application

- Information Extraction
- Speech Recognition
- POS Tagging
- Shallow Parsing
- Handwriting Recognition
- Protein Secondary Structure Prediction
- Video Analysis
- Facial Expression Dynamic Modelling

Information Extraction

- Mengidentifikasi frasa pada bahasa yang mengacu ke tipe entitas dan relasi tertentu pada teks.
- Named entity recognition adalah pekerjaan mengidentifikasi nama orang, tempat, organisasi, dll pada teks

  people organizations places

  – Michael Dell is the CEO of Dell Computer Corporation and lives in Austin Texas.
- Mengekstraksi informasi yang relevan terhadap aplikasi tertentu, misalnya pada iklan mobil bekas :

  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

- Pelabelan rangkaian juga berguna dalam rangkaian genetik pada analisis genome.

  extron intron

  – AGCTAACGTTCGATACGGATTACAGCCT

Problems with Sequence Labeling as Classification

- Tidak mudah mengintegrasikan informasi dari kategori token pada kanan dan kirinya.
- Sulit untuk mempropagasi ketidakpastian antar keputusan dan menentukan mana penetapan kategori yang paling tepat dari token pada rangkaian tersebut.

Probabilistic Sequence Models

- Model rangkaian probabilistik (probabilistic sequence models) memperbolehkan integrasi ketidakpastian dari beberapa klasifikasi yang interdependen dan secara kolektif menentukan mana assignment global yang paling tepat.
- Dua buah model yang menjadi standard :
  - Hidden Markov Model (HMM)
  - Conditional Random Field (CRF)

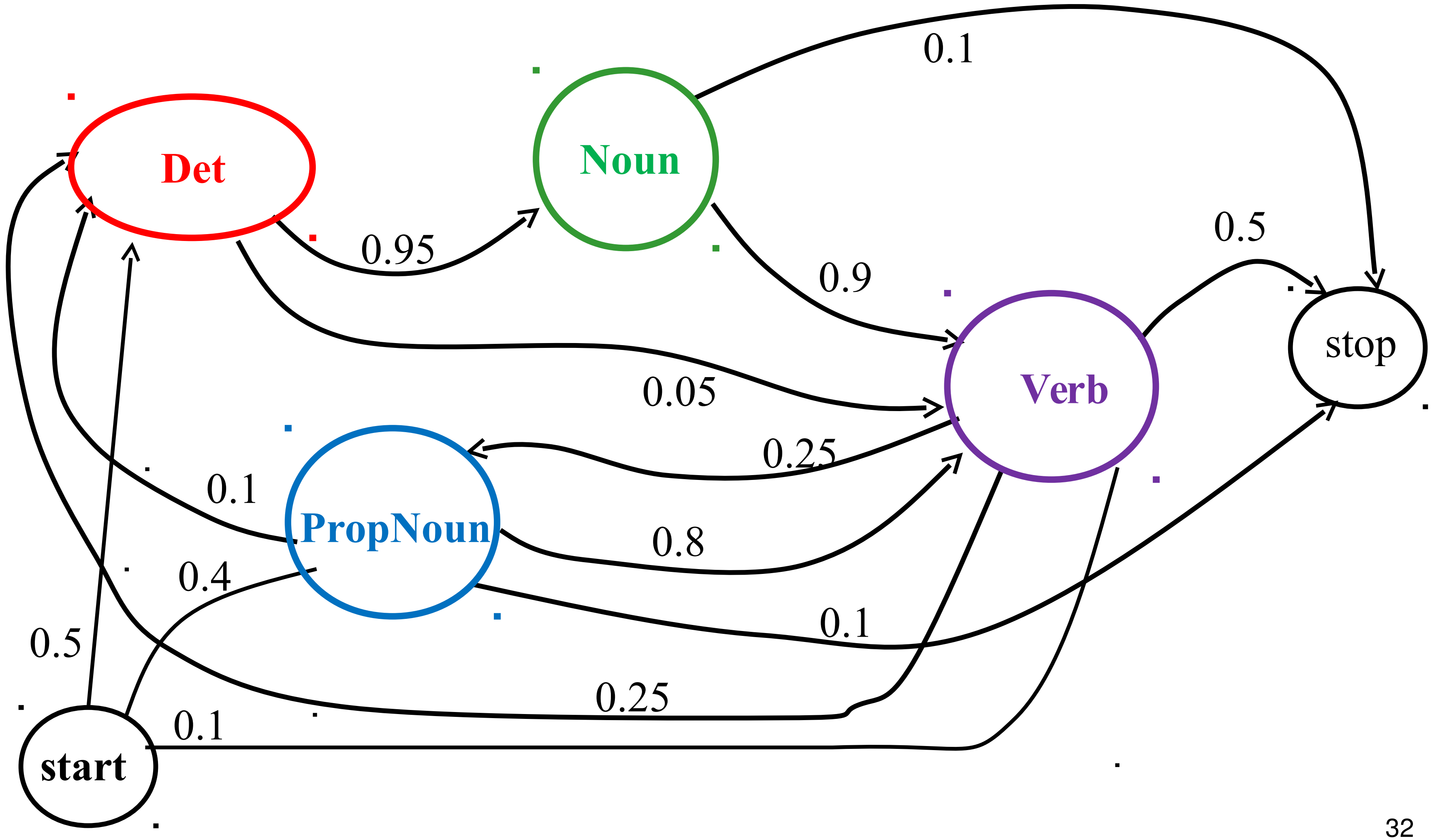
Markov Model / Markov Chain

- Sebuah mesin finite state dengan transisi state berupa probabilistik.
- Diasumsikan bahwa state berikutnya hanya bergantung dari state saat ini dan independen terhadap apa yang terjadi sebelumnya (Markov assumption).

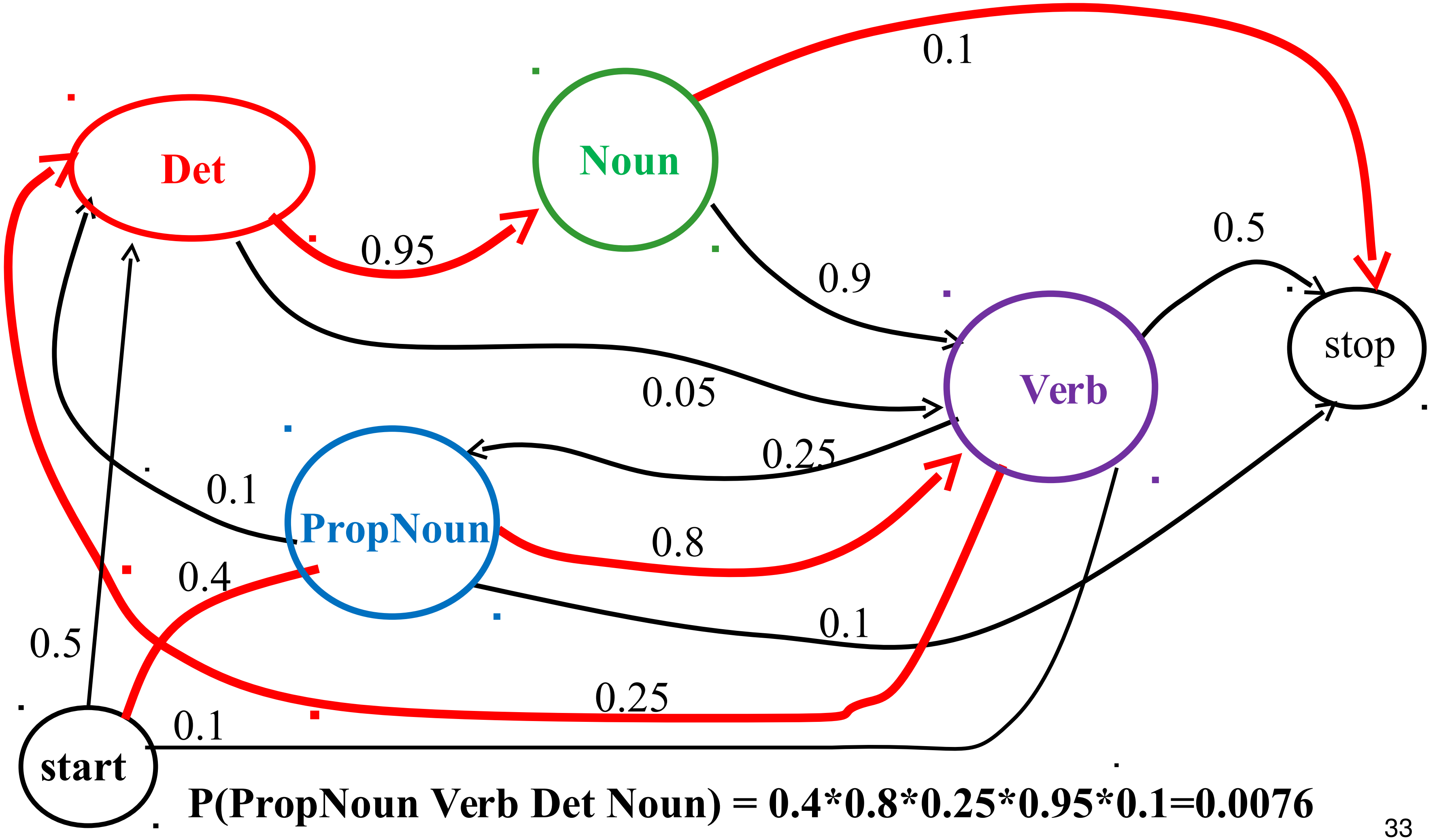
Markov Model / Markov Chain

- Observasi $x_{1:T}$ dimodelkan dengan state machine (tersembunyi) yang menggeneratenya (generative model)
- State y_t berkorespondensi dengan label, state sequencenya adalah $y_{1:T}$
- Sejumlah hingga label diperbolehkan , $y_t \in Y$ dimana $|Y|$ berhingga
- Markov assumption $p(y_t|y_1,y_2,\ldots,y_{t-2},y_{t-1}) = p(y_t|y_{t-1})$
- Transisi dari state y_{t-1} ke state lain y_t berlangsung setiap saat
- Observasi x_t dimunculkan setelah transisi
- Parameter model :
  - Probabilitas transisi antar state
  - Probabilitas kemunculan emisi dari observasi state
  - Probabilitas mulai dari state tertentu

Sample Markov Model for POS



Sample Markov Model for POS



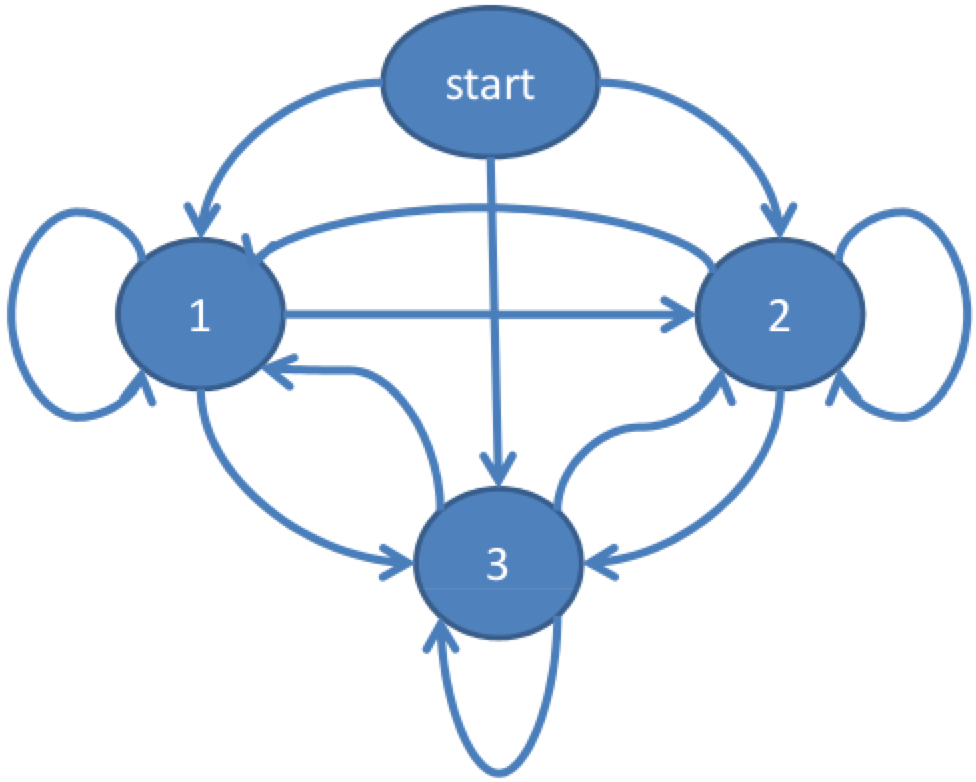
P(PropNoun Verb Det Noun) = 0.4*0.8*0.25*0.95*0.1=0.0076

Hidden Markov Model

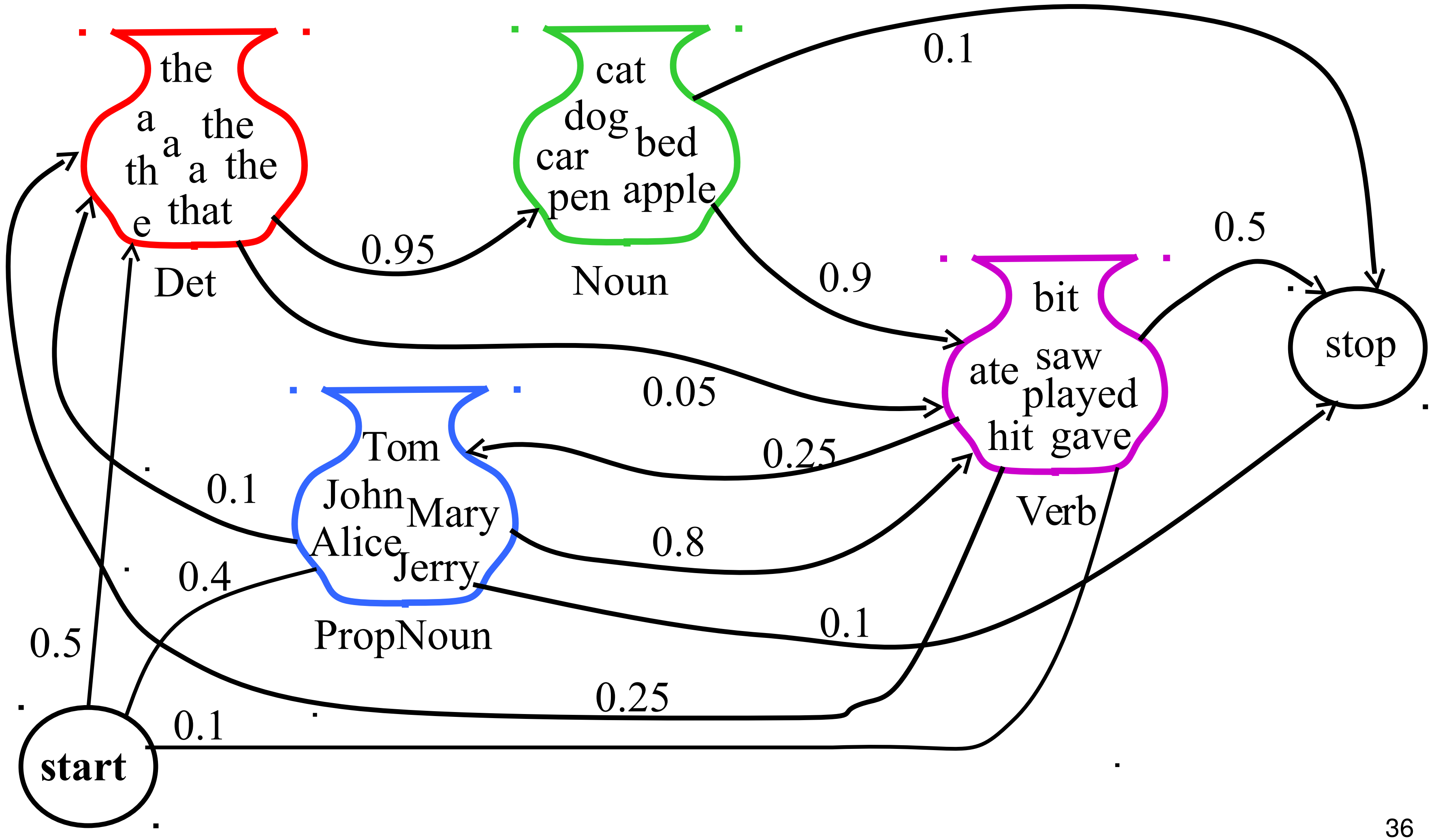
- HMM bisa ditampilkan dalam beberapa bentuk :
  - State transition diagram
  - Graphical model
  - Trellis / Lattice diagram

State Transition Diagram

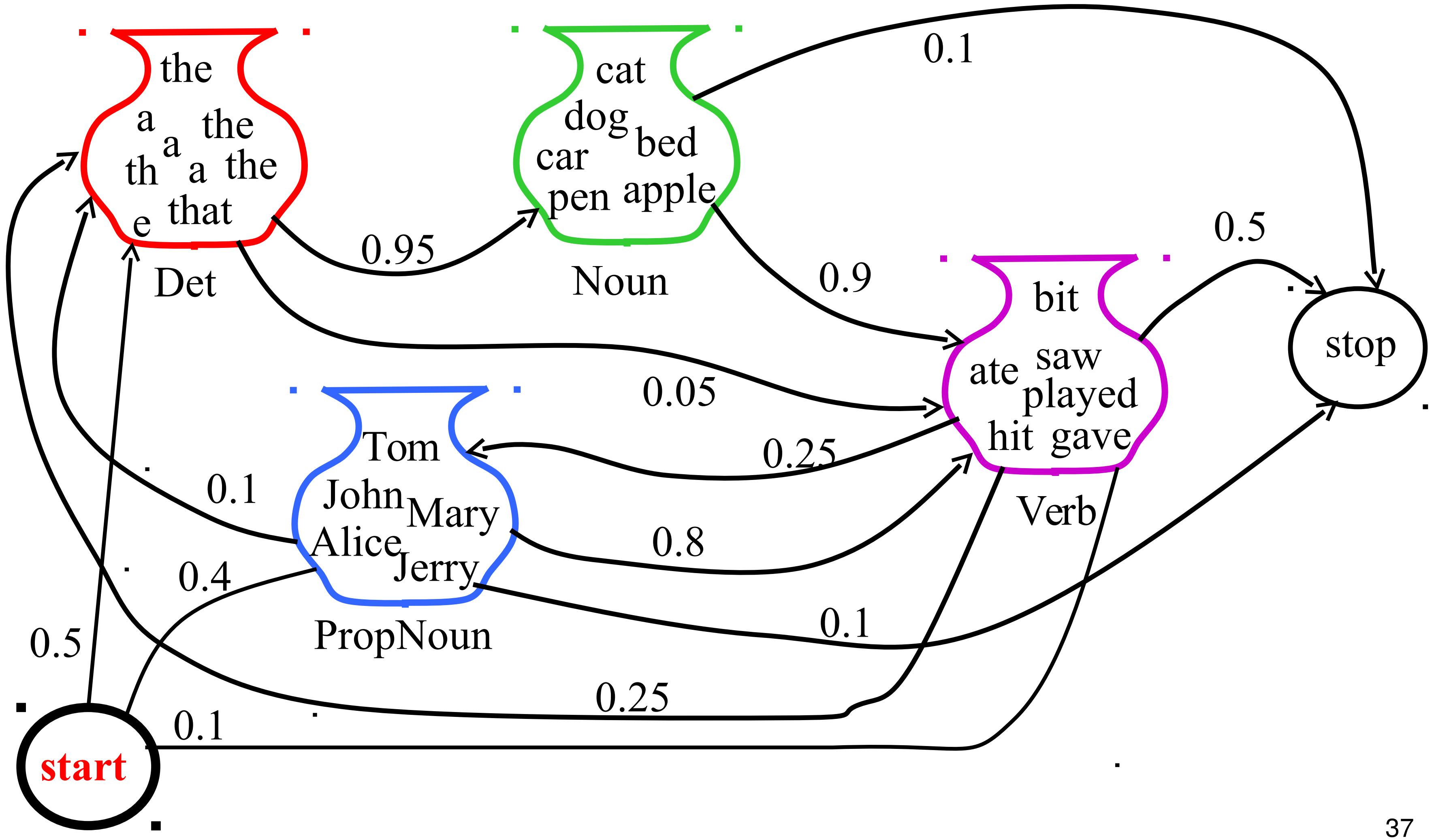
- Observasi tidak dituliskan secara eksplisit dalam diagram ini
- Transisi yang tidak mungkin (peluangnya 0) tidak dituliskan



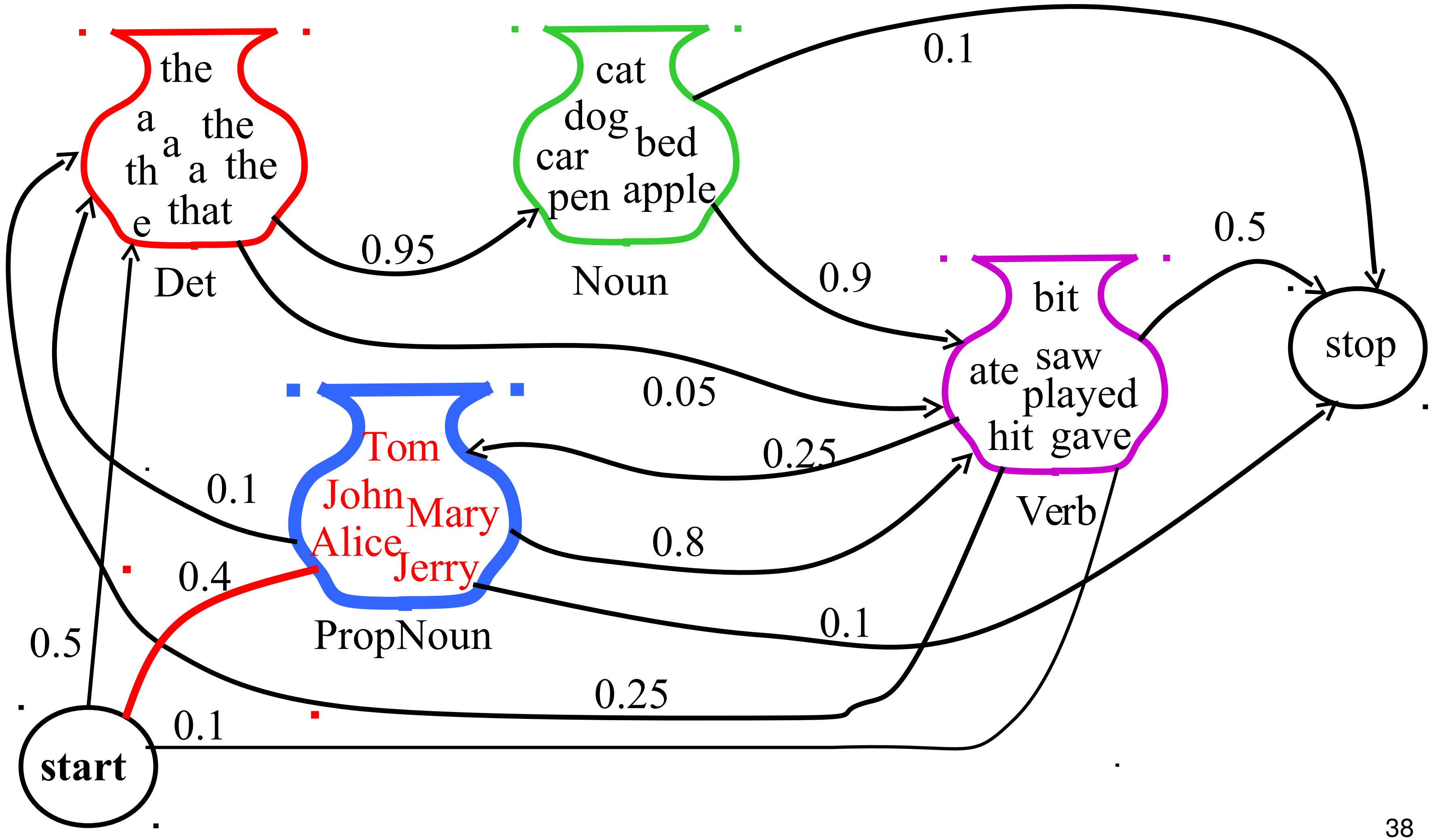
Sample HMM for POS



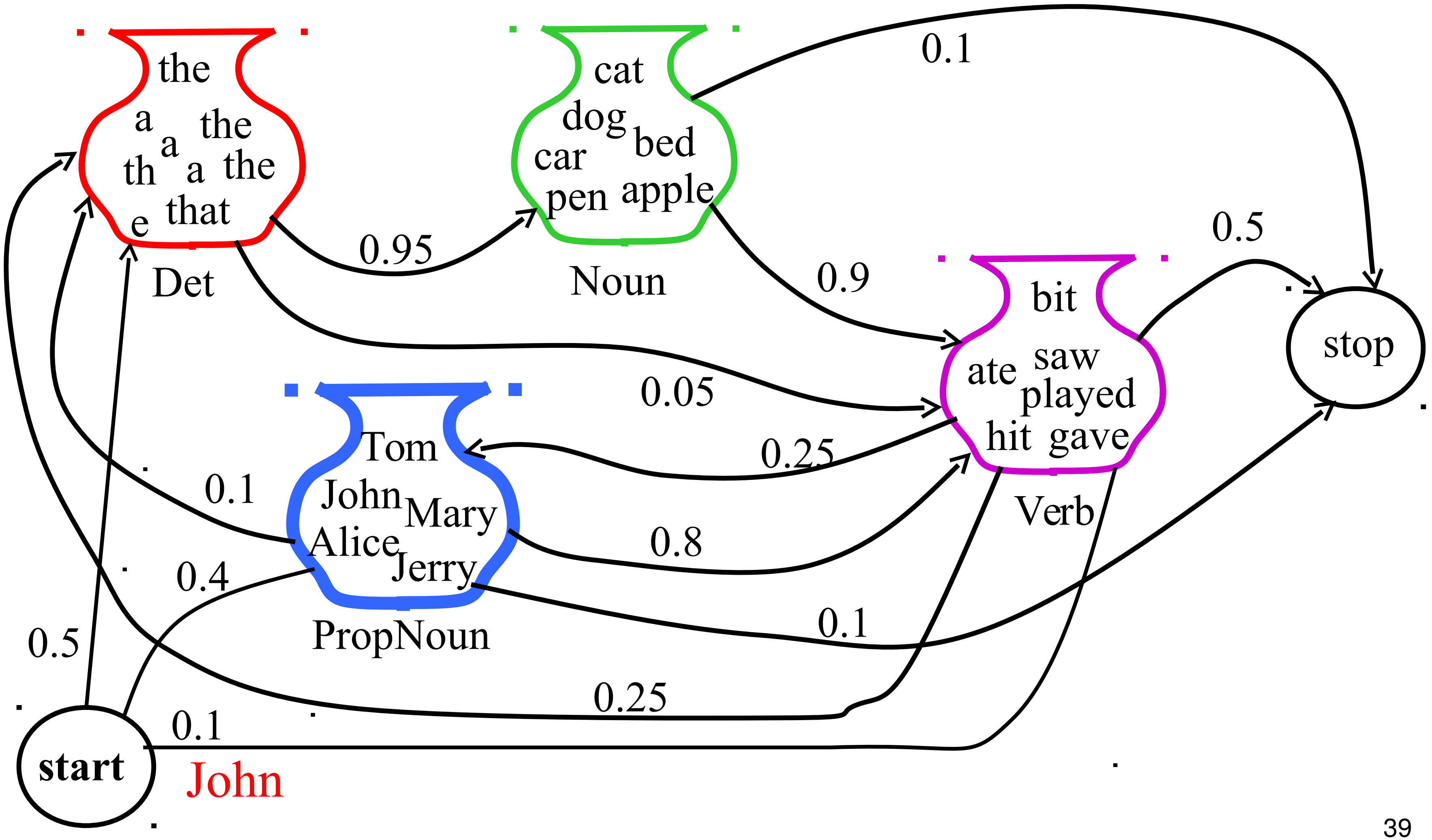
Sample HMM Generation



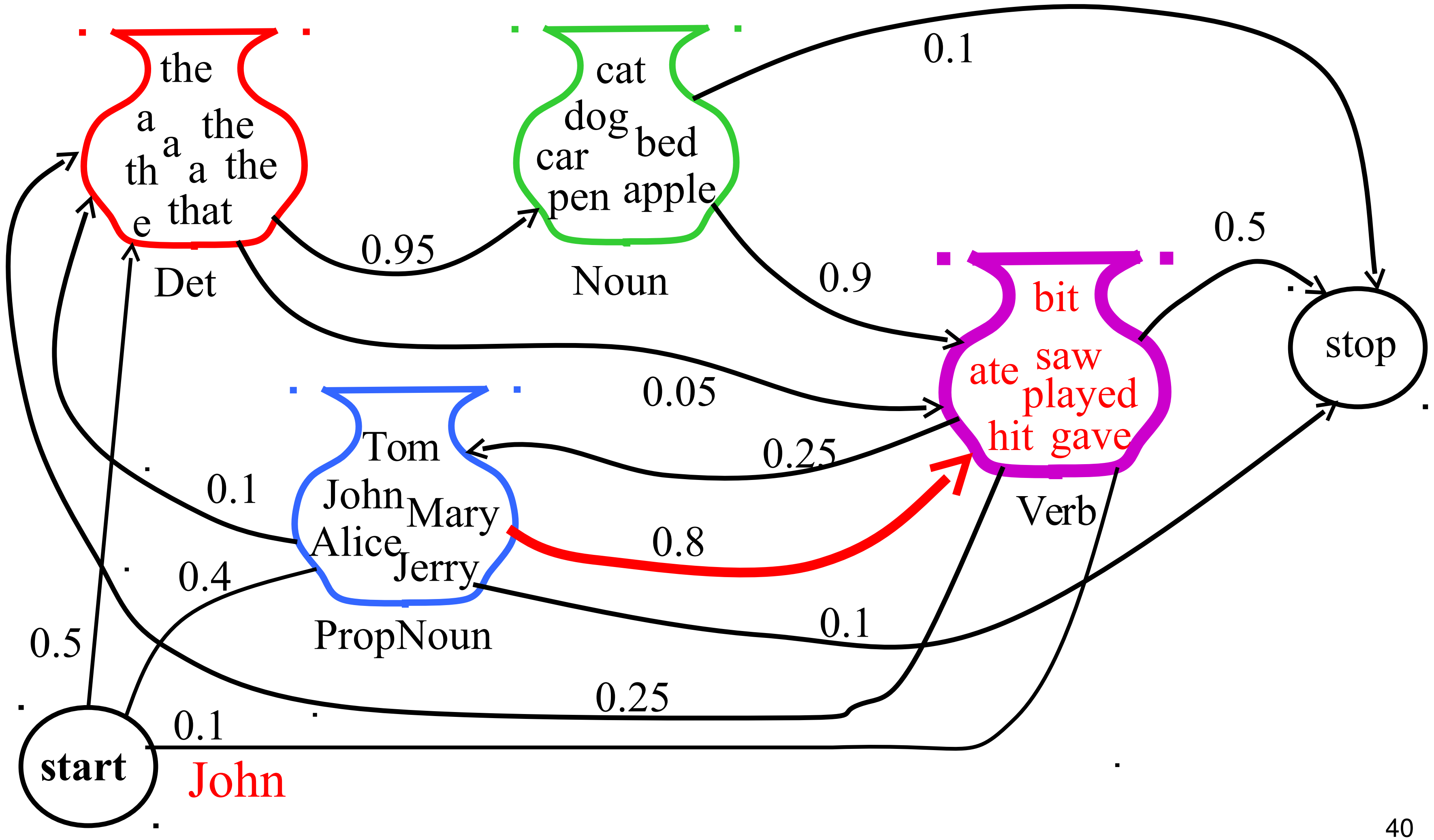
Sample HMM Generation



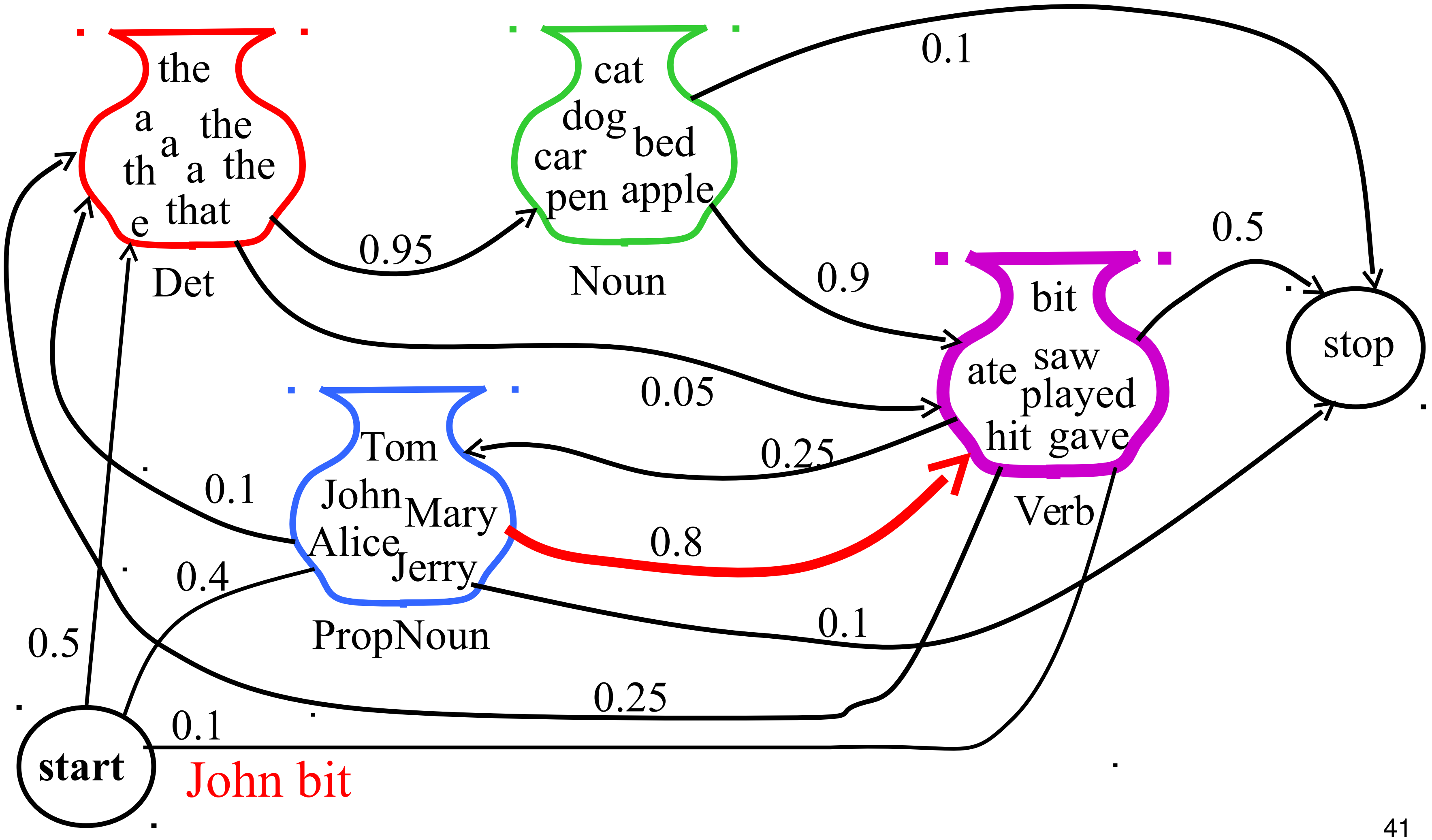
Sample HMM Generation



Sample HMM Generation

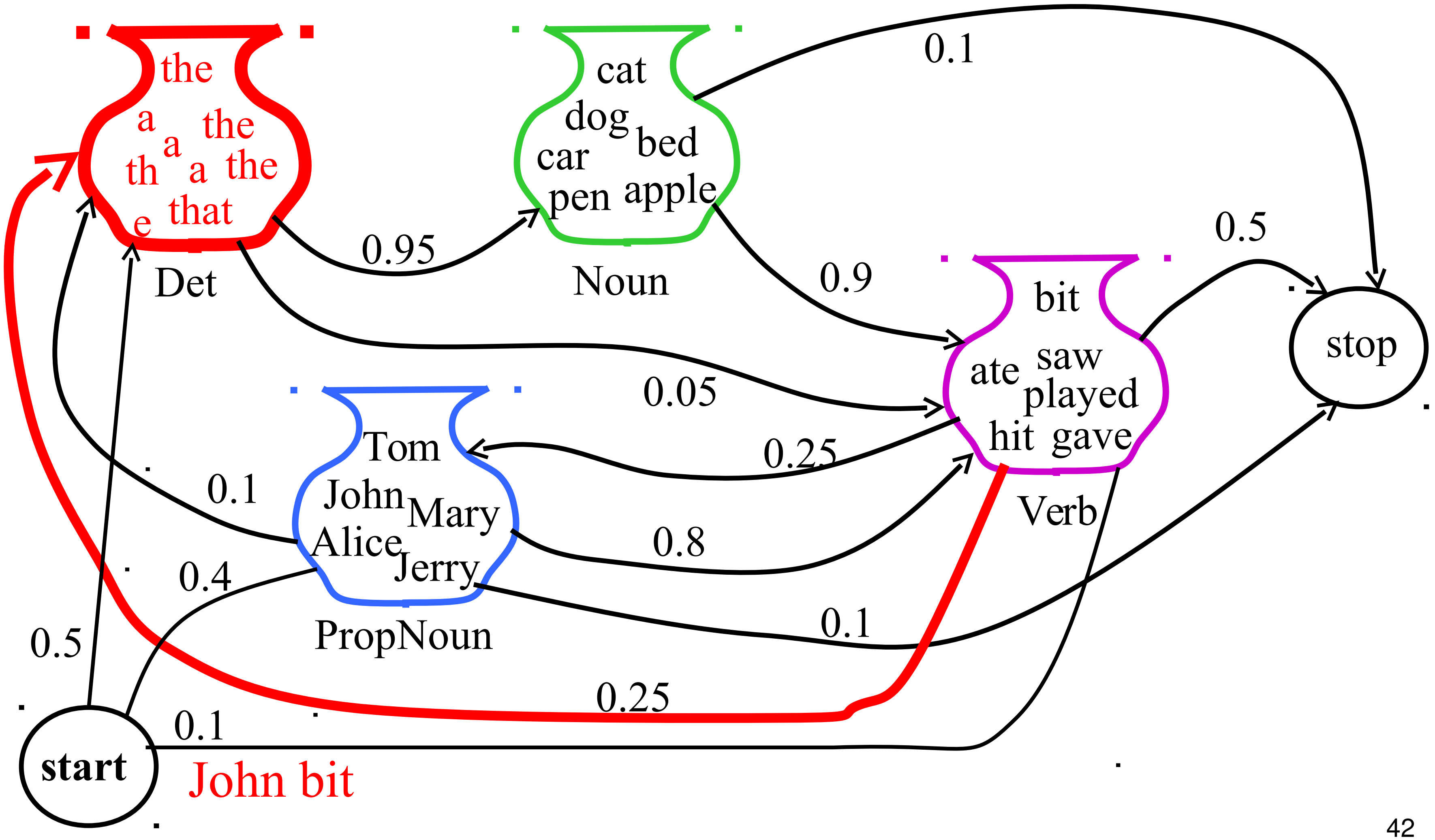


Sample HMM Generation

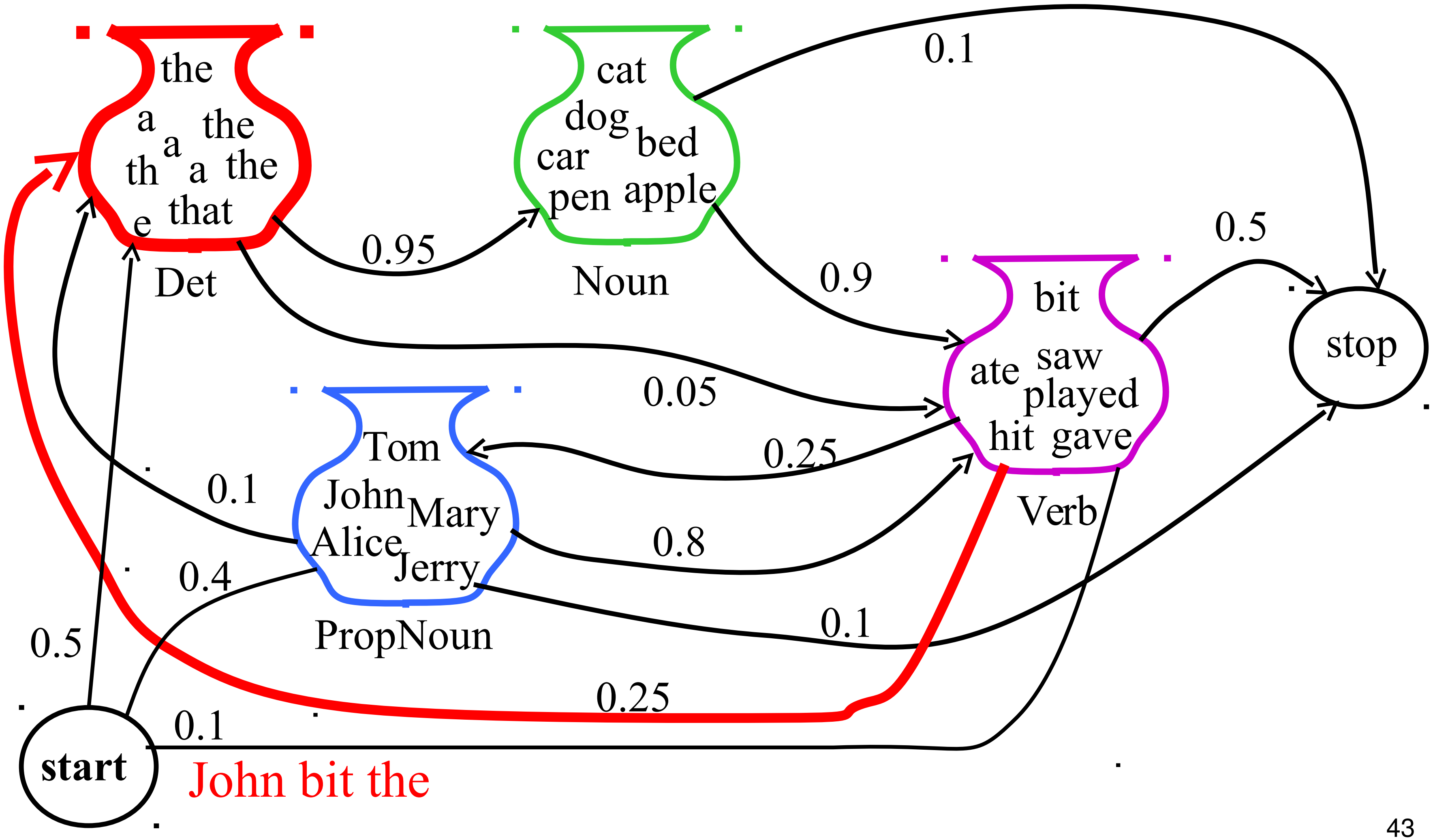


John bit

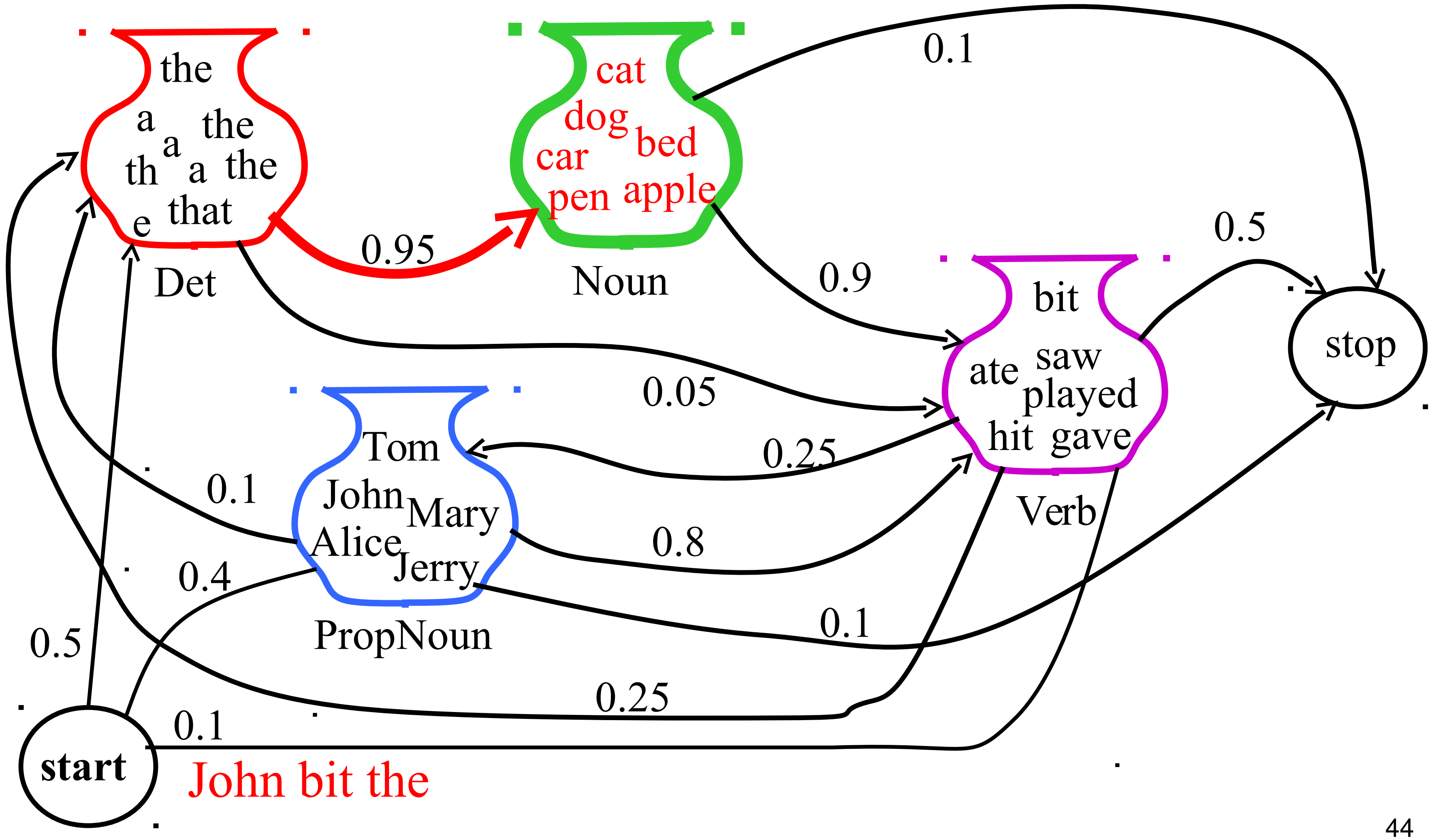
Sample HMM Generation



Sample HMM Generation

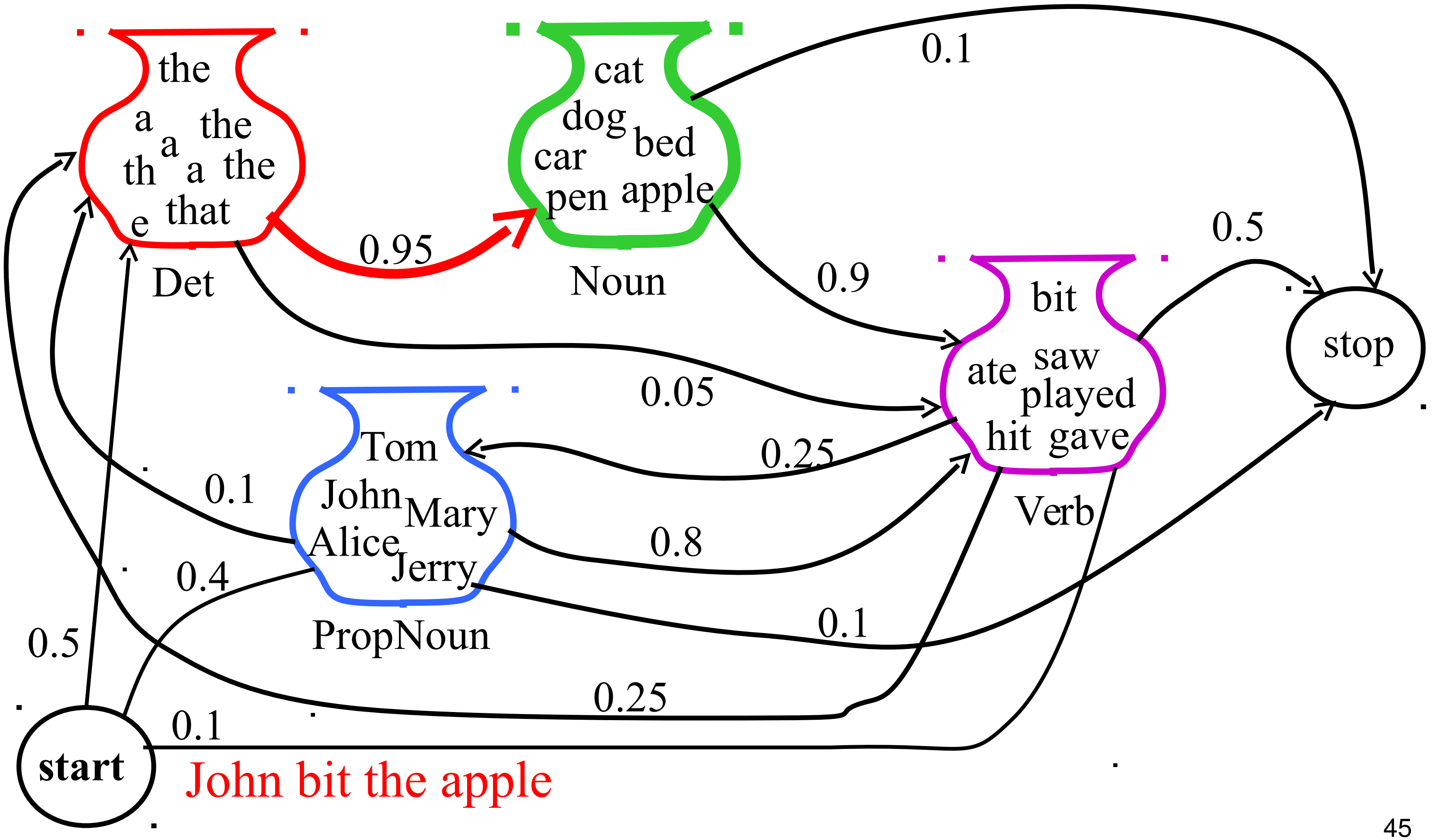


Sample HMM Generation



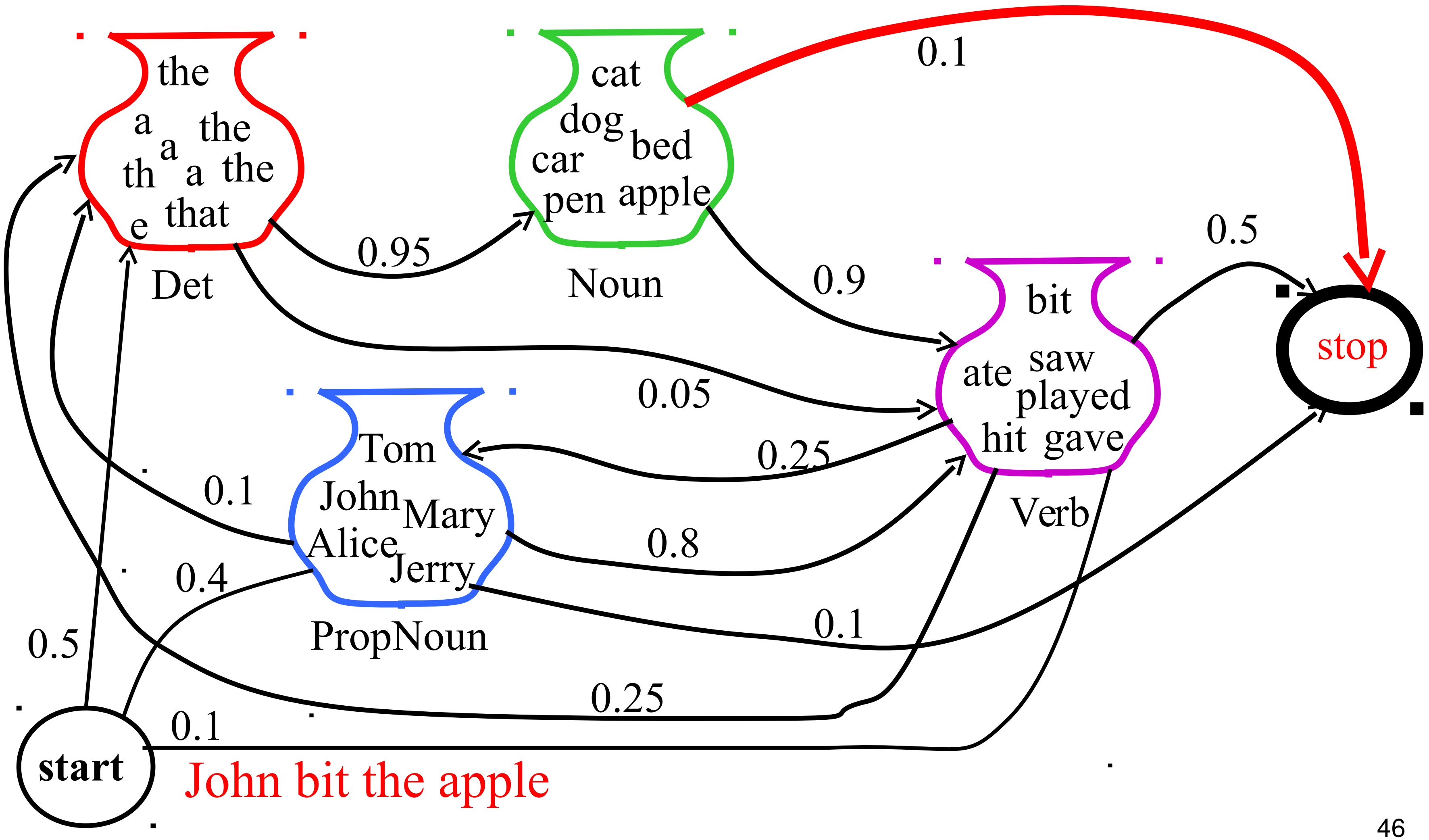
John bit the

Sample HMM Generation

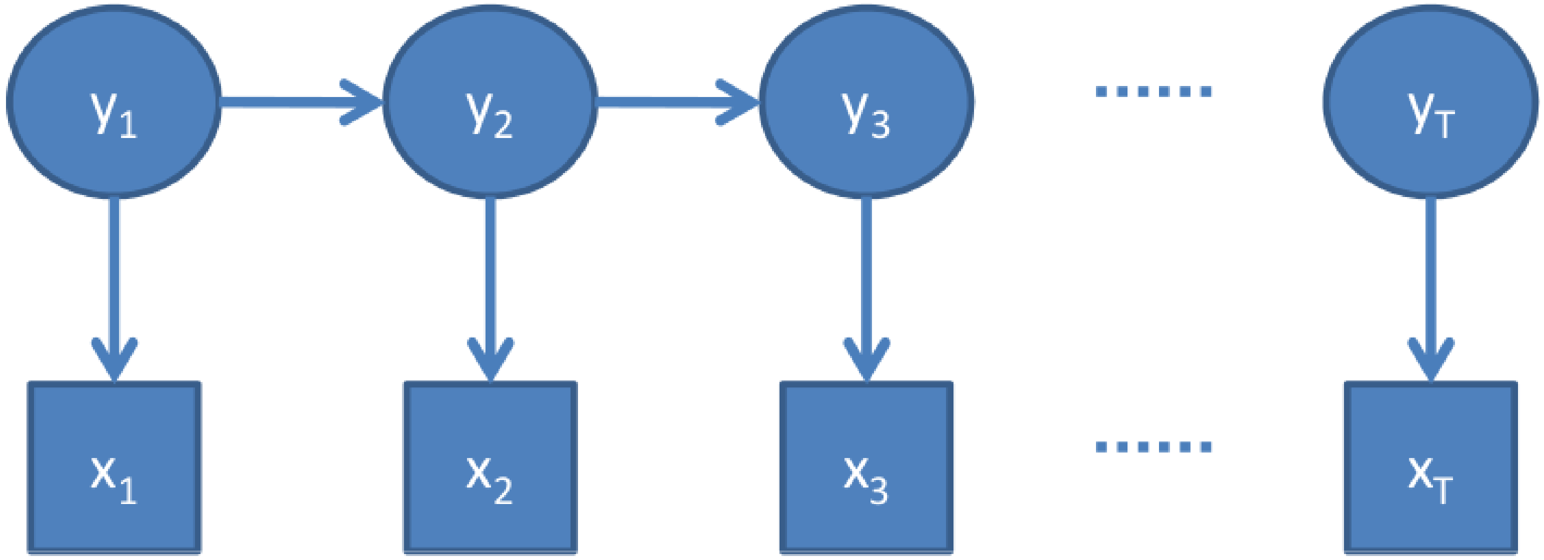


John bit the apple

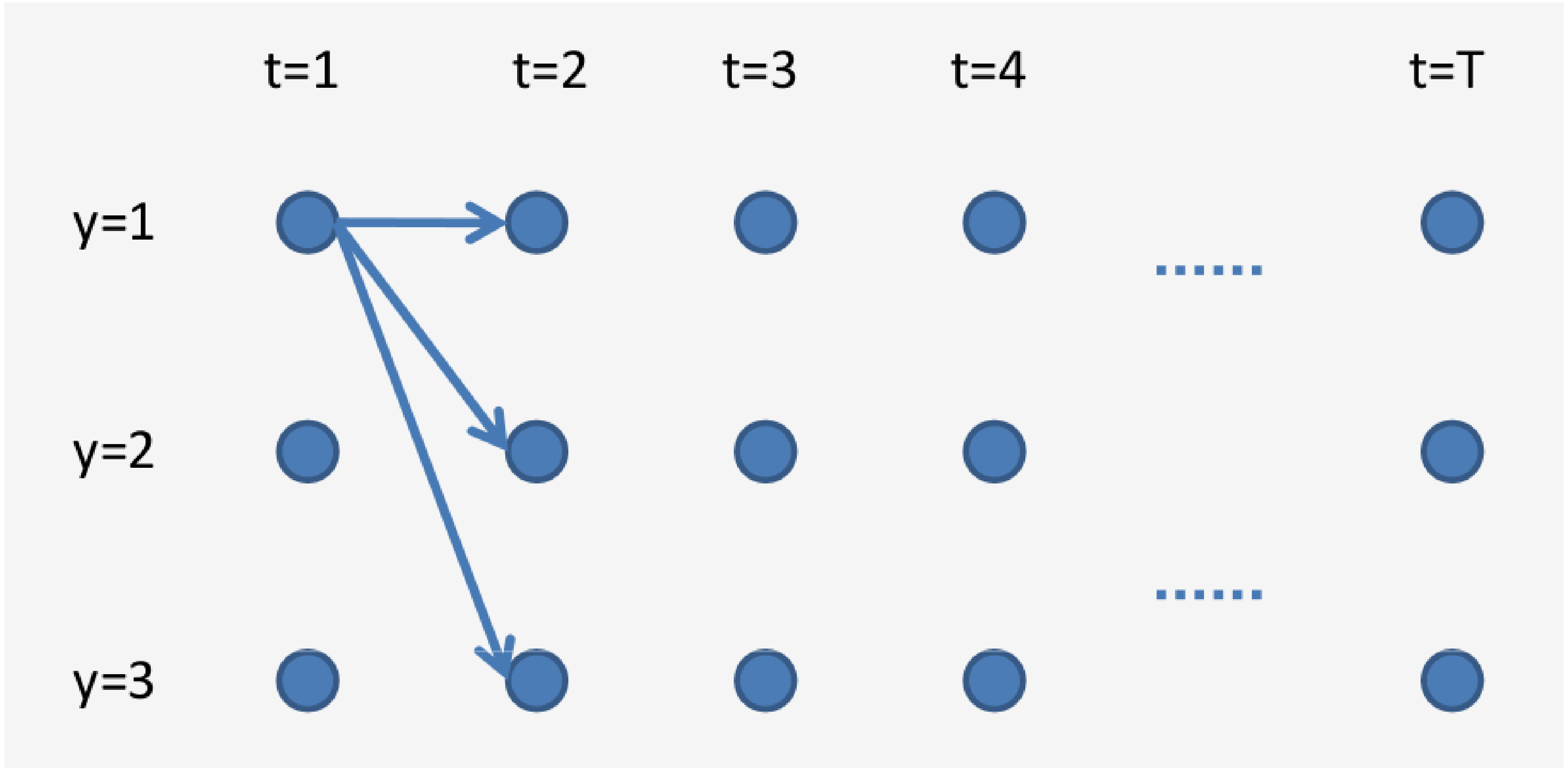
Sample HMM Generation



Graphical Model



Trellis / Lattice



- Observasi tidak ditampilkan, label ditampilkan secara eksplisit

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

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

- 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 $b_{qt}(k)$
```

Three Useful HMM Tasks

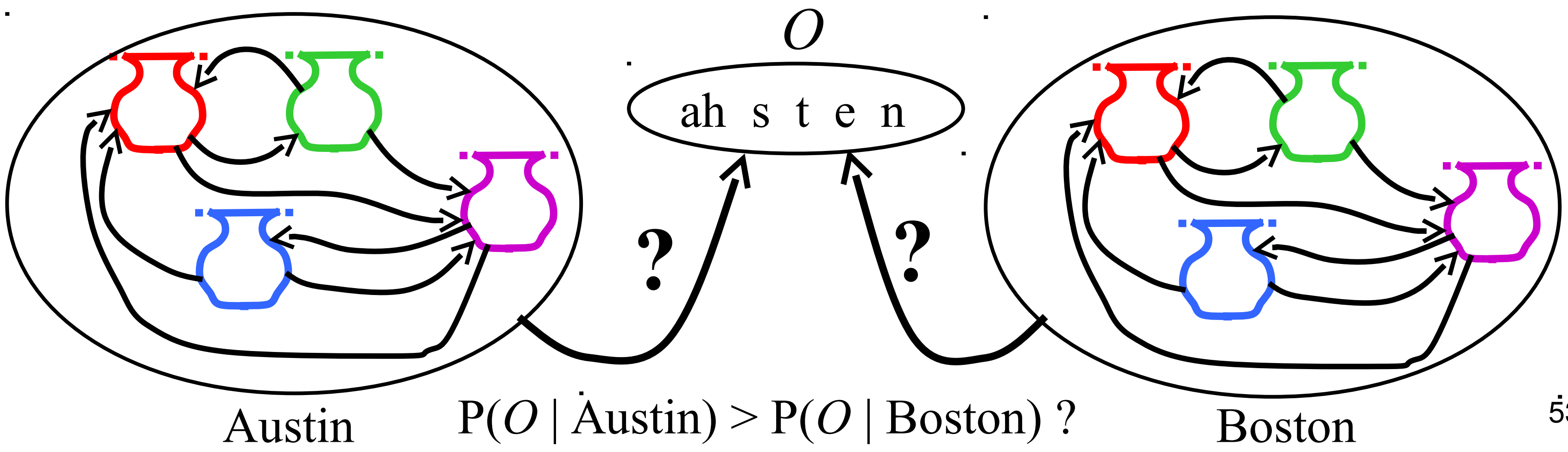
- Observation Likelihood: To classify and order sequences.
  - Diberikan sebuah model λ, kita ingin mengevaluasi peluang barisan observasi tertentu $O = \{O_1 O_2 ... O_T\}$, yaitu $P(O|\lambda)$
- Most likely state sequence (Decoding): To tag each token in a sequence with a label.
  - Diberikan sebuah model λ dan barisan observasi O, kita ingin mengetahui barisan label $Q = \{q_1 q_2 ... q_T\}$ yang memiliki peluang tertinggi untuk menghasilkan O
  - Dengan kata lain, mencari Q^* yang memaksimalkan $P(Q|\, O,\lambda)$
- Maximum likelihood training (Learning): To train models to fit empirical training data.
  - Diberikan training set dari barisan observasi $X = \{O^k\}_k$, kita ingin menghasilkan model yang memaksimalkan peluang yang menghasilkan X
  - Dengan kata lain, kita ingin menemukan λ^* yang memaksimalkan $P(X|\lambda)$

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

Sequence Classification

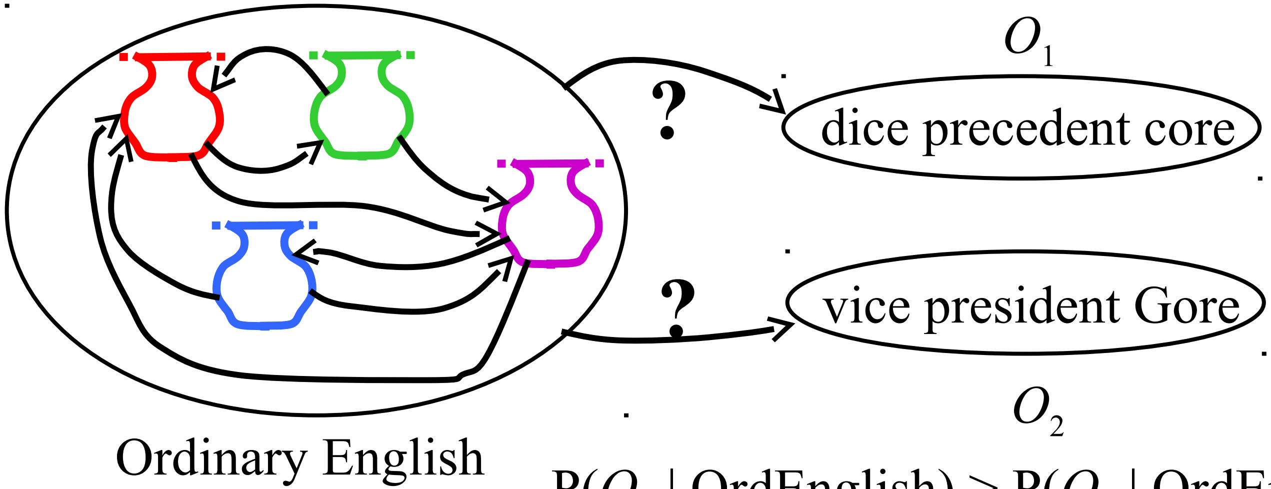
- Assume an HMM is available for each category (i.e. language).
- What is the most likely category for a given observation sequence, i.e. which category's HMM is most likely to have generated it?
- Used in speech recognition to find most likely word model to have generate a given sound or phoneme sequence.



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.



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

HMM: Observation Likelihood Naïve Solution

- Diberikan barisan observasi $O = \{O_1O_2...O_T\}$ dan barisan state $Q = \{q_1q_2...q_T\}$, peluang dari observasi O jika diberikan Q adalah

  $$P(O \mid Q,\lambda) = \Pi\, P(O_t \mid q_t,\lambda) = b_{q1}(O_1).b_{q2}(O_2)\ \dots\ b_{qT}(O_T)$$

- Tapi tidak bisa dihitung karena kita tidak tahu barisan statenya. Peluang dari barisan state Q adalah

  $$P(Q \mid \lambda) = P(q_1)\ \Pi\ P(q_t|q_{t-1})$$

- Sehingga peluang gabungannya

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

- Untuk mendapatkan $P(O| \lambda)$, kita perlu menjumlahkan peluang $P(O,Q \mid \lambda)$ terhadap seluruh Q yang mungkin

  $$P(O \mid \lambda) = \Sigma\ P(O,Q \mid \lambda)$$

- Tidak praktikal karena ada N^T macam Q yang mungkin sehingga kompleksitas komputasinya berordo $O(TN^T)$.

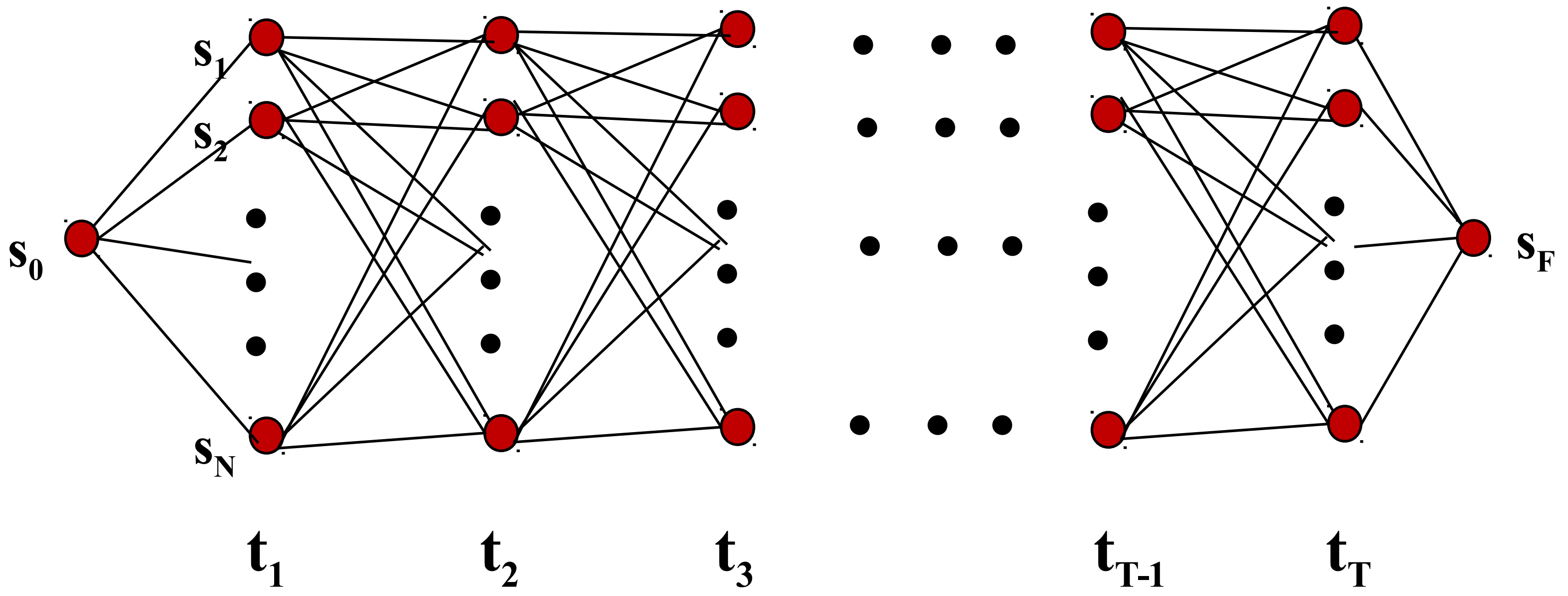
HMM: Observation Likelihood Naïve Solution

- 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 $P(O,Q| \lambda) = P(O| Q, \lambda)\, P(Q| \lambda)$.
- Sum this over all possible state sequences to get $P(O| \lambda)$.
- Computationally complex: $O(TN^T)$.

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.

Forward Trellis



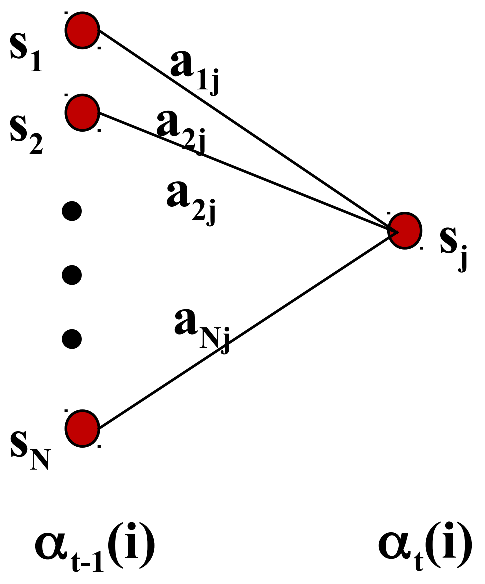
- Continue forward in time until reaching final time point and sum probability of ending in final state.

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).

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

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

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

- 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

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

Computing Forward Probabilities

- Contoh, misalnya diketahui model dengan dua buah state S_1 dan S_2 dengan fungsi transisi sebagai berikut :

Transisi Dari \ Ke	S1	S2
S1	0.6	0.4
S2	0.3	0.7

Initial Transition Probability Matrix $A=\{a_{i,j}\}$

Computing Forward Probabilities

- Dengan output probabilities dan initial state probabilities sbb :

Output Prob.	R	W	B
S1	0.3	0.4	0.3
S2	0.4	0.3	0.3

Output Probabilities $B=\{b_j(k)\}$

Initial State Prob	
S1	0.8
S2	0.2

Initial State Probabilities

Computing Forward Probabilities

- Pertanyaan :
  - Ingin diketahui peluang dari kalimat RWBB
- Langkah pengerjaan :
  - Initialization
  - Dihitung terlebih dahulu peluang masing masing state pada waktu awal (t=1) dengan output R

$\alpha_1(j) = a_{0j}\, b_j(o_1)$

$\alpha_1(1) = a_{01}\, b_1(R) = 0.8 * 0.3 = 0.24$

$\alpha_1(2) = a_{02}\, b_2(R) = 0.2 * 0.4 = 0.08$

Computing Forward Probabilities

- Recursion
- Pada waktu selanjutnya, dihitung dari waktu sebelumnya, increment dari waktu t=1

$$\alpha_t(j) = [\Sigma\, \alpha_{t-1}(i)\, a_{ij}]\, b_j(o_t)$$

$$\alpha_2(1) = [\alpha_1(1)\, a_{11} + \alpha_1(2)\, a_{21}]\, b_1(W)$$

$$= [0.24 * 0.6 + 0.08 * 0.3] * 0.4$$

$$= 0.0672$$

$$\alpha_2(2) = ?$$

Computing Forward Probabilities

– Recursion

$$\alpha_t(j) = [\Sigma\ \alpha_{t-1}(i)\ a_{ij}]\ b_j(o_t)$$

$$\alpha_3(1) = [\alpha_2(1)\ a_{11} + \alpha_2(2)\ a_{21}]\ b_1(B)$$

$$= [0.0672*0.6 + 0.0456*0.3]*0.3$$

$$=$$

$$\alpha_3(2) = ?$$

Computing Forward Probabilities

– Recursion

Sekarang sudah didapatkan nilai peluang dari forward probabilities HMM

	R	W	B	B
S1	0.24	0.0672	0.0162	0.0045
S2	0.08	0.0456	0.0176	0.0056

Computing Forward Probabilities

- Termination

$$P(O|\lambda) = \alpha_{T+1}\,(S_F) = \Sigma\,\alpha_T(i)$$

$$\begin{aligned} P(O|\lambda) &= \alpha_5\,(S_F) = \Sigma\,\alpha_4(i)\,a_{i1} \\ &= \alpha_4(1)\,a_{11} + \alpha_4(2)\,a_{21} \\ &= 0.0045 + 0.0056 \\ &= 0.0101 \end{aligned}$$

Computing Forward Probabilities

Kadang, setelah input tersebut diketahui, kita ingin mengetahui hal lain, misalnya peluang berada di state i pada waktu t dengan input tersebut.

- Contoh : Peluang berada di state S_1 pada waktu t=3 dengan input RWB ?

$$P(Q_3=S_1|RWB) = P(Q_3=S_1,RWB)/P(RWB)$$

- Pembilangnya adalah forward probabilities, tapi bagaimana dengan penyebutnya? Itu adalah jumlah semua peluang pada waktu t=3.

Computing Forward Probabilities

	R	W	B	B
S1	0.24	0.0672	0.0162	0.0045
S2	0.08	0.0456	0.0176	0.0056
Σ	0.32	0.1128	0.0338	0.0101

$$P(Q_3{=}S_1|RWB) = P(Q_3{=}S_1,RWB)/P(RWB)$$

$$= 0.0162 / 0.0338$$

$$= 0.4787$$

$$\gamma_3(1) = P(Q_3{=}S_1|RWB) = 0.4787$$

- $\gamma_t(i)$ didefinisikan sebagai peluang berada di state S_i pada waktu t, diberikan O dan λ

Forward Computational Complexity

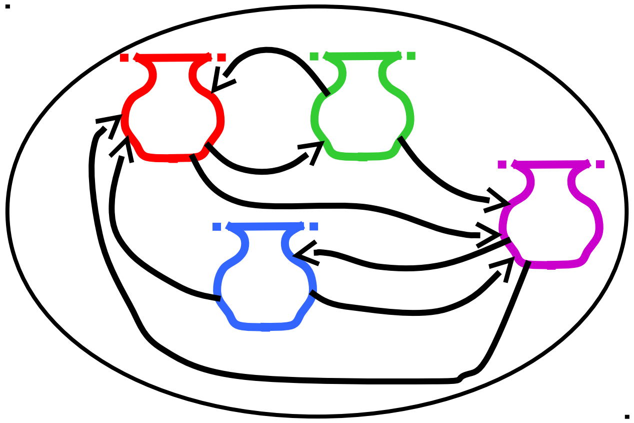
- Requires only $O(TN^2)$ time to compute the probability of an observed sequence given a model.
- Exploits the fact that all state sequences must merge into one of the N possible states at any point in time and the Markov assumption that only the last state effects the next one.

Backward

- Optional

Most Likely State Sequence (Decoding)

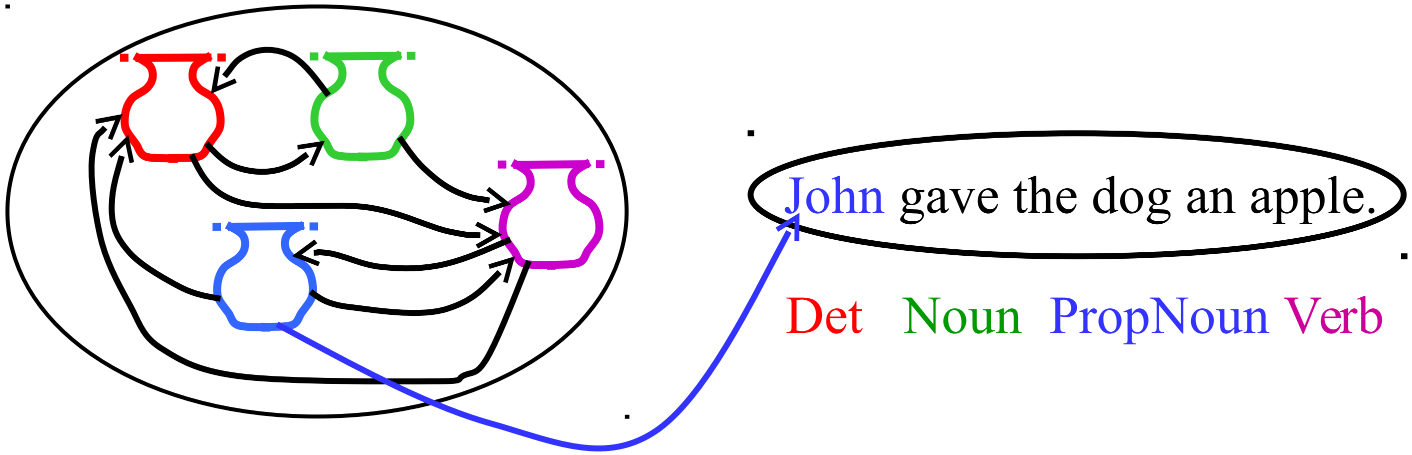
- Given an observation sequence, O, and a model, λ, what is the most likely state sequence, $Q = q_1, q_2, \ldots q_T$, that generated this sequence from this model?
- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens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



John gave the dog an apple.

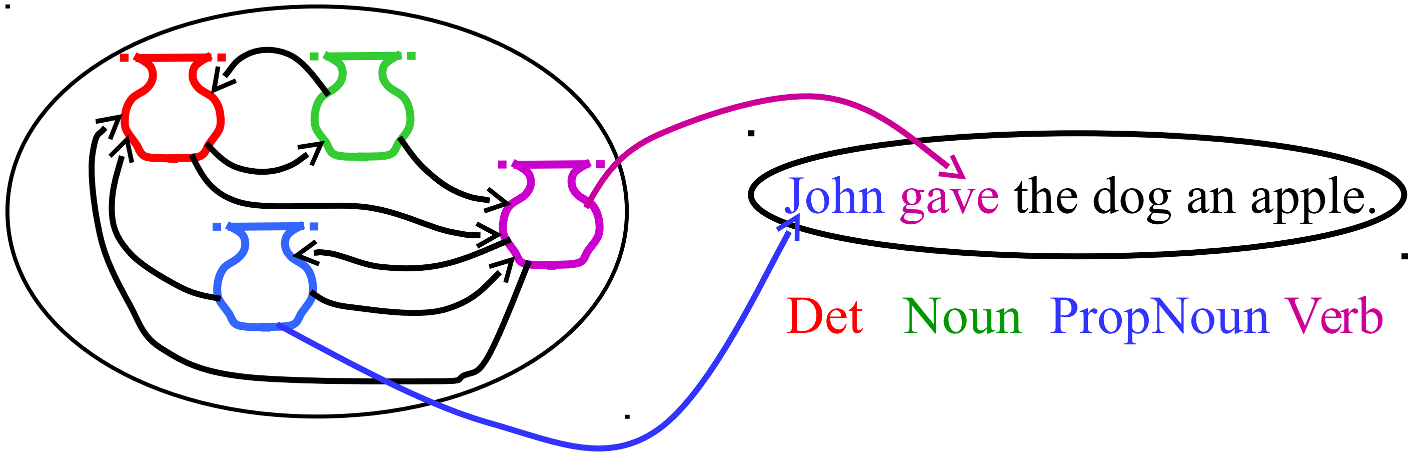
Most Likely State Sequence

- 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?
- 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.



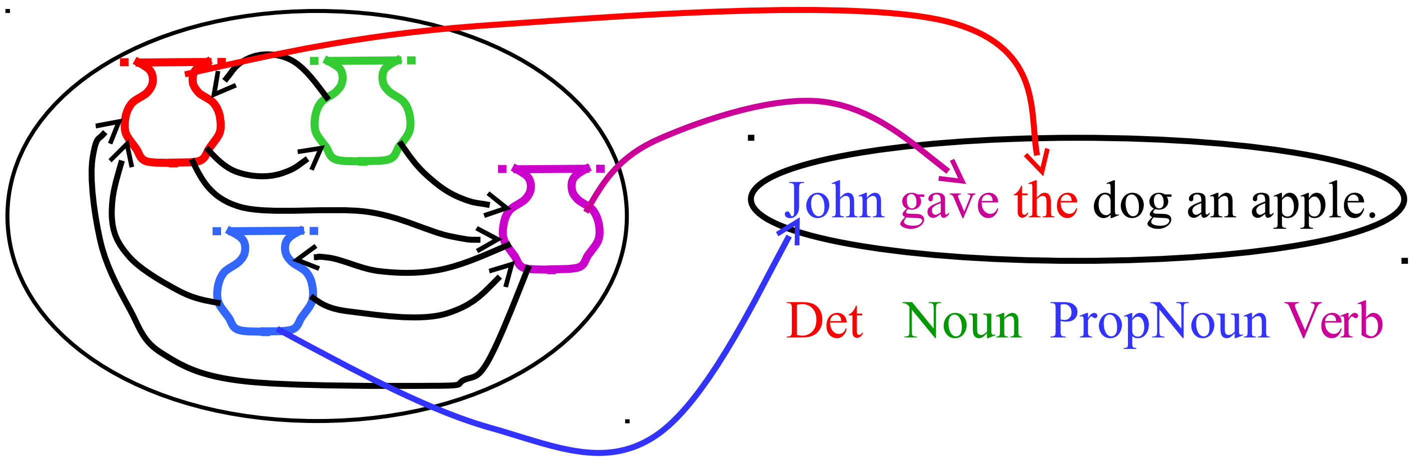
Most Likely State Sequence

- 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.



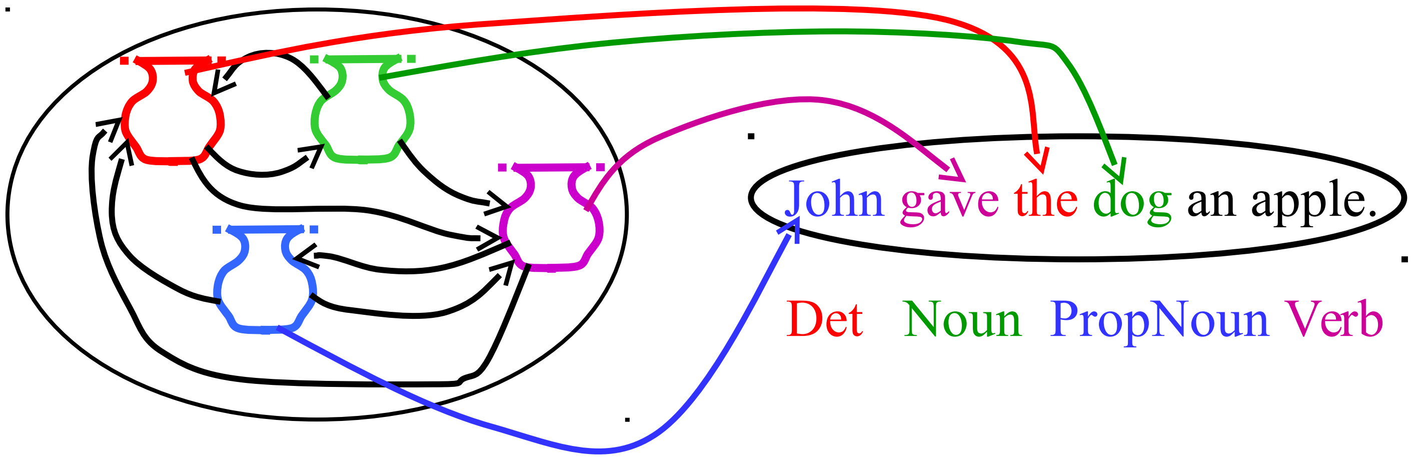
Most Likely State Sequence

- 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.



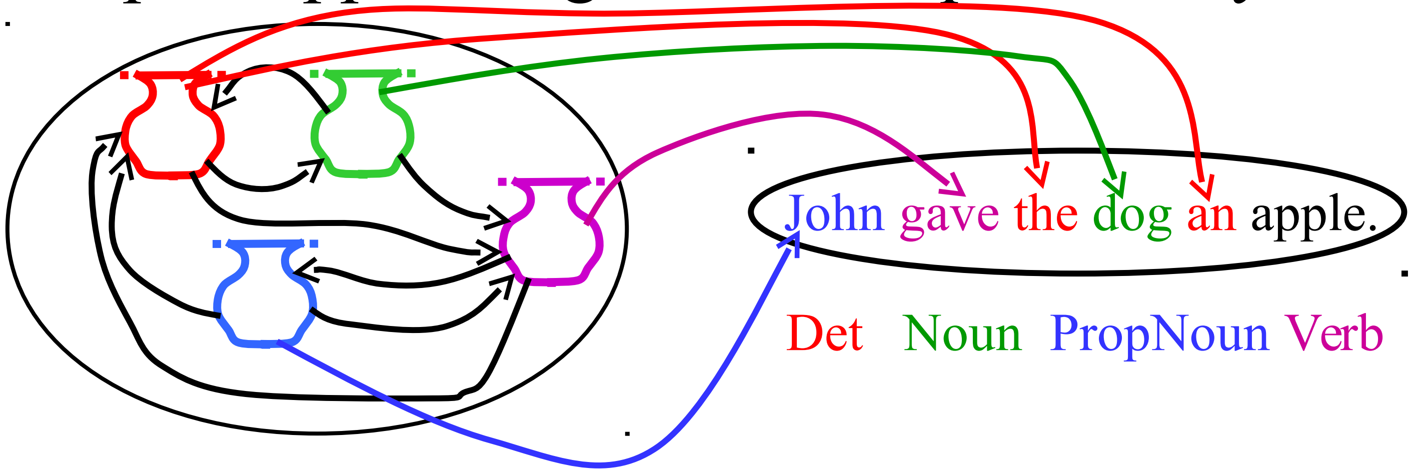
Most Likely State Sequence

- 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.



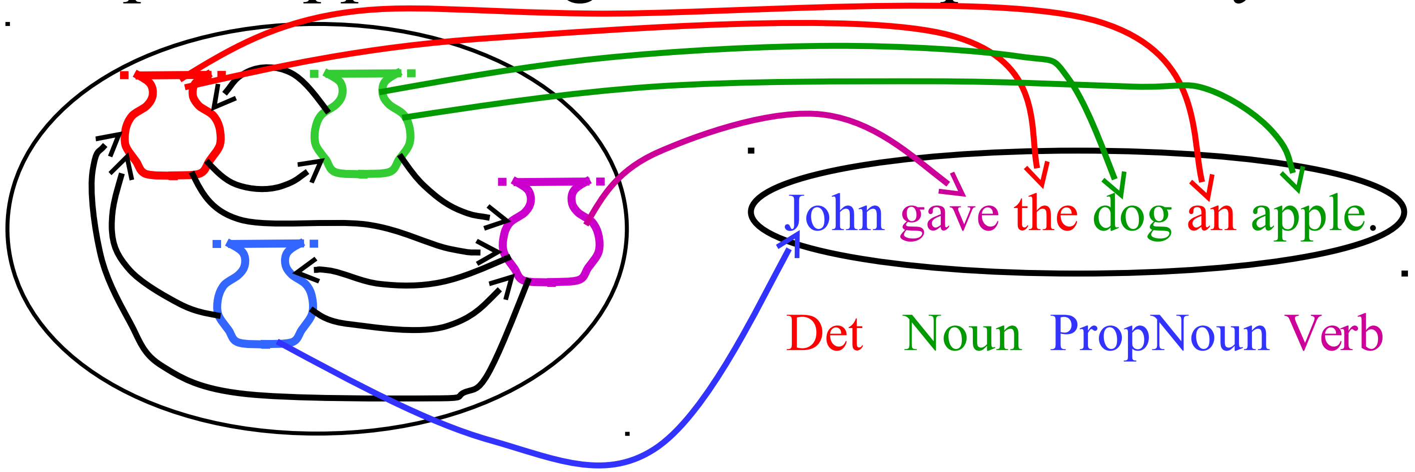
Most Likely State Sequence

- 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.



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.



HMM: Most Likely State Sequence Efficient Solution

- Kita bisa menggunakan naïve algorithm dengan cara mengamati semua kemungkinan barisan state dengan panjang T.
- Dynamic Programming juga bisa dimanfaatkan untuk mengeksploitasi asumsi markov dan menentukan dengan efisien barisan state yang paling mungkin jika diberikan model dan observasi.
- Prosedur standardnya disebut Viterbi algorithm (Viterbi, 1967) dan memiliki kompleksitas O(TN^2) juga.

Viterbi Scores

- Menghitung secara rekursif peluang barisan state yang paling mungkin dengan mempertimbangkan bahwa *t* buah observasi pertama berakhir di state s$_j$

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

$$v_t(j) = \max_{q_0, q_1, \ldots q_{t-1}} P(q_0, q_1, \ldots q_{t-1}, \; o_1, \ldots o_{t-1}, \; q_t =$$

- Memiliki "backpointers". Optimal path bisa dibaca dengan backtracking dari *T* dengan memilih yang paling mungkin di tiap langkahnya.

Computing the Viterbi Scores

- Initialization

  $v_1(j) = a_{0j}\, b_j(o_1) \qquad 1 \le j \le N$

- Recursion

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

- Termination

  $p^* = \max(\, v_T(i)\,)$

mirip dengan Forward algorithm, tapi operatornya **max**, bukan **sum**

Computing the Viterbi Backpointers

- Initialization

  $\Psi_1(j) = s_0 \qquad 1 \leq j \leq N$

- Recursion

  $\Psi_t(j) = \text{argmax } v_{t-1}(i)\, a_{ij} \qquad 1 \leq j \leq N, 1 \leq t \leq T$

- Termination

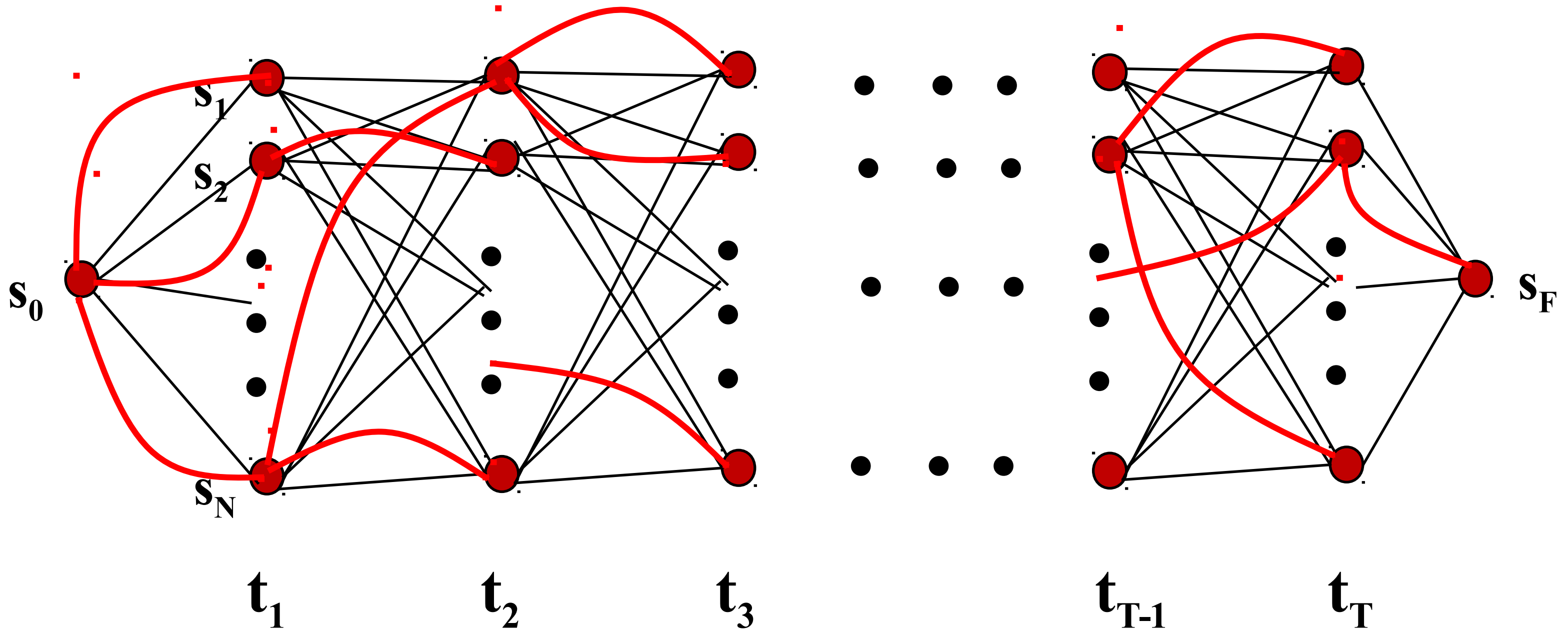
  $q^*_T = \text{argmax } v_T(i)$

- Backtracking

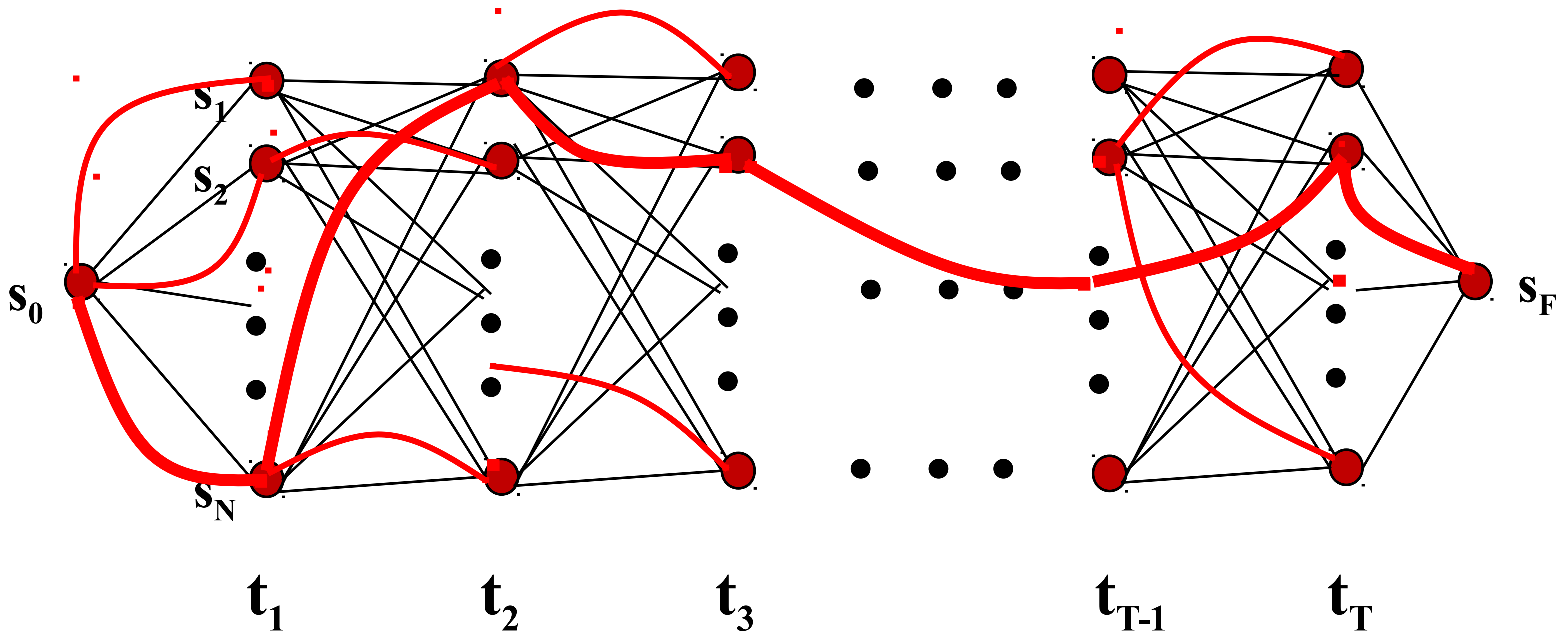
  $q^*_t = \Psi_{t+1}(q^*_{t+1}) \qquad t = T\text{-}1, T\text{-}2, \ldots, 1$

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

Viterbi Backpointers



Viterbi Backtrace



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

Computing the Viterbi Scores

- Model yang sama dengan yang digunakan pada forward probabilities sebagai berikut :

Transisi Dari \ Ke	S1	S2
S1	0.6	0.4
S2	0.3	0.7

Initial Transition Probability Matrix A={$a_{i,j}$}

Computing the Viterbi Scores

- Dengan output probabilities dan initial state probabilities sbb :

Output Prob.	R	W	B
S1	0.3	0.4	0.3
S2	0.4	0.3	0.3

Output Probabilities B={b_j(k)}

Initial State Prob	
S1	0.8
S2	0.2

Initial State Probabilities

Computing the Viterbi Scores

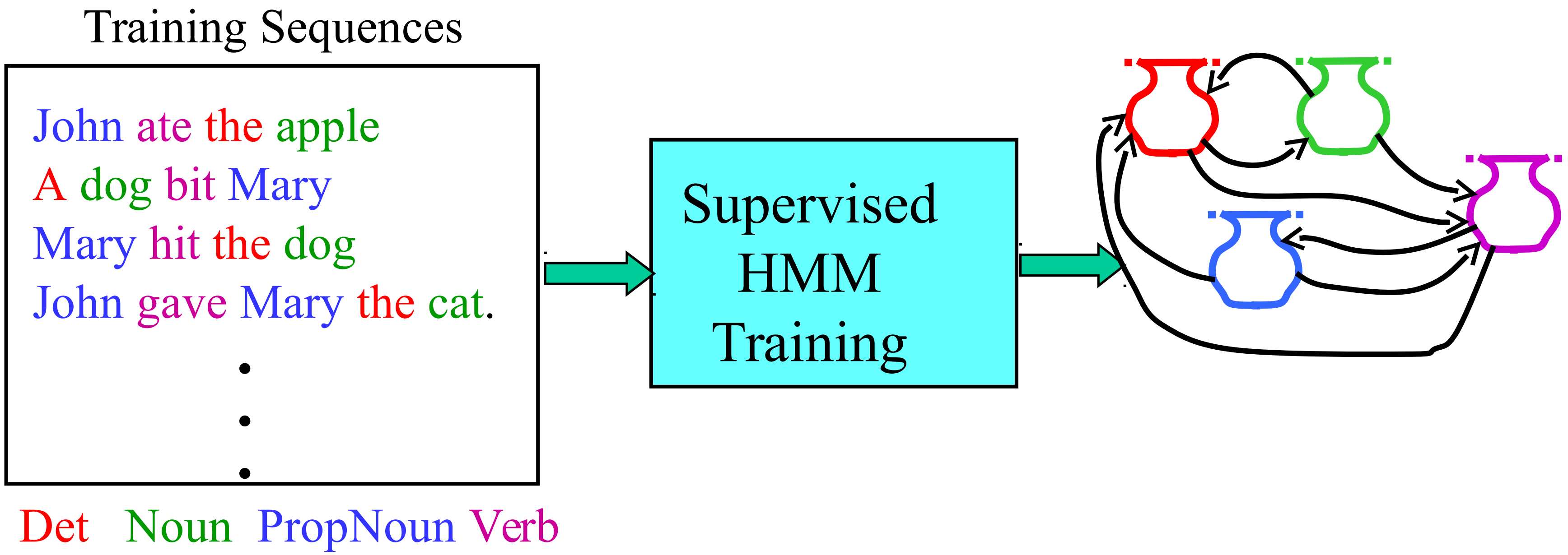
- Jika diketahui observasinya adalah RWBB, rangkaian state apa yang paling mewakili kalimat tersebut?
- Kemungkinannya ada 16 macam. Yang mana yang paling besar peluangnya?
  - $S_1S_1S_1S_1$
  - $S_1S_1S_1S_2$
  - …
  - $S_2S_2S_2S_2$

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.



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

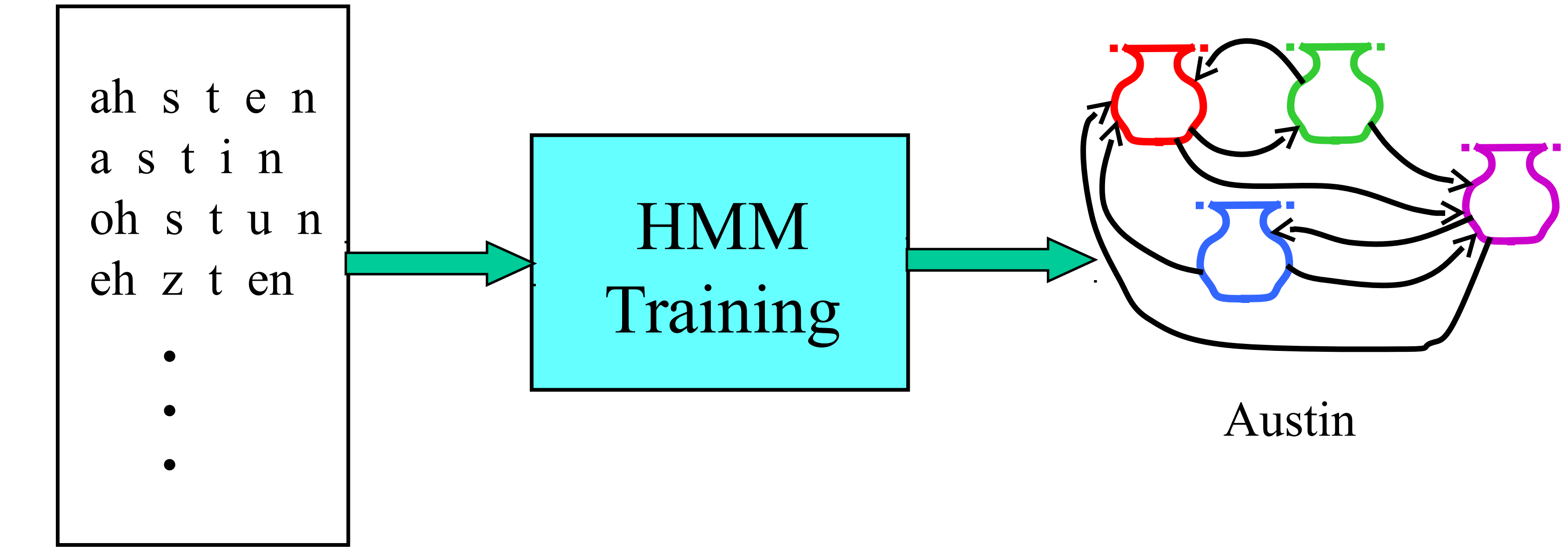
- Use a corpus of labeled sequence data to easily construct an HMM using supervised training.
- Given a novel unlabeled test sequence to tag, use the Viterbi algorithm to predict the most likely (globally optimal) tag sequence.

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



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.

Bayes Theorem

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

Simple proof from definition of conditional probability:

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

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

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

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

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.

EM Algorithm

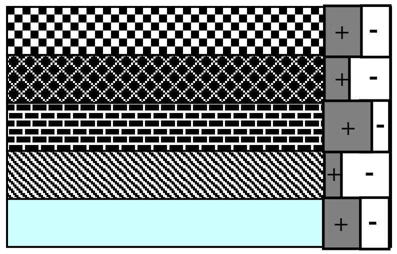
- Iterative method for learning probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories.
- Learn an initial probabilistic model by estimating model parameters θ from this randomly labeled data.
- Iterate following two steps until convergence:
  - 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

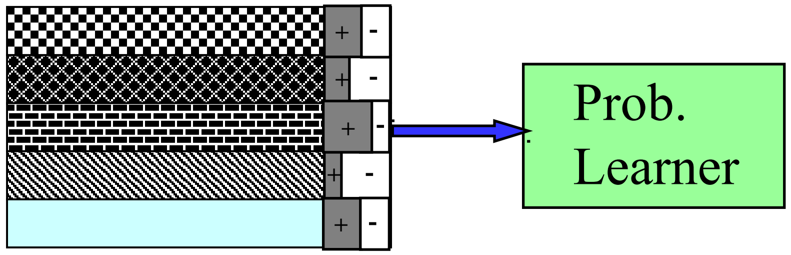
Unlabeled Examples



EM

Initialize:

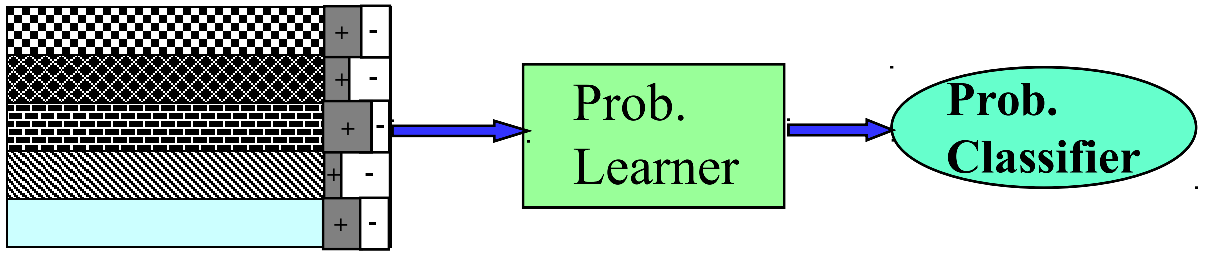
Give soft-labeled training data to a probabilistic learner



EM

Initialize:

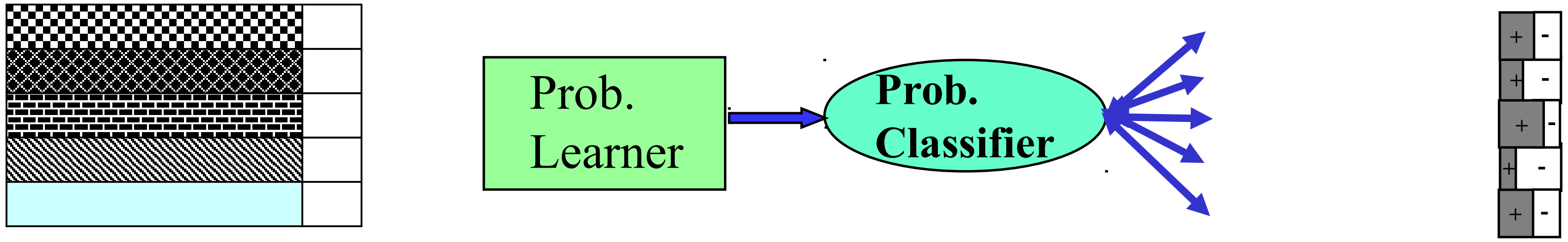
Produce a probabilistic classifier



EM

E Step:

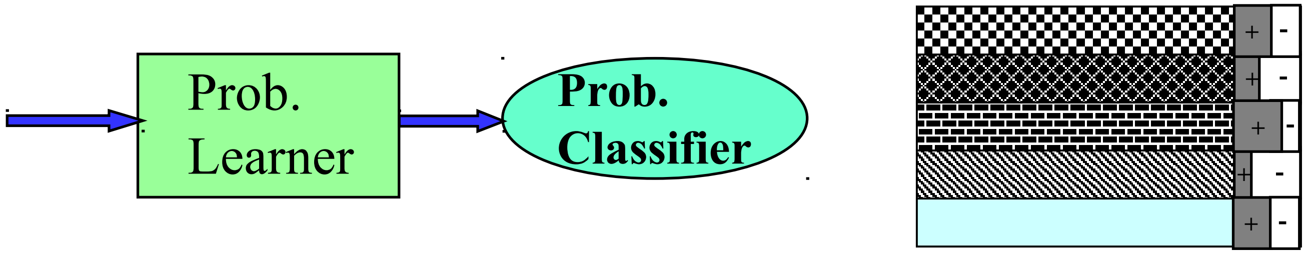
Relabel unlabled data using the trained classifier



EM

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

See textbook for detailed equations for E and M steps

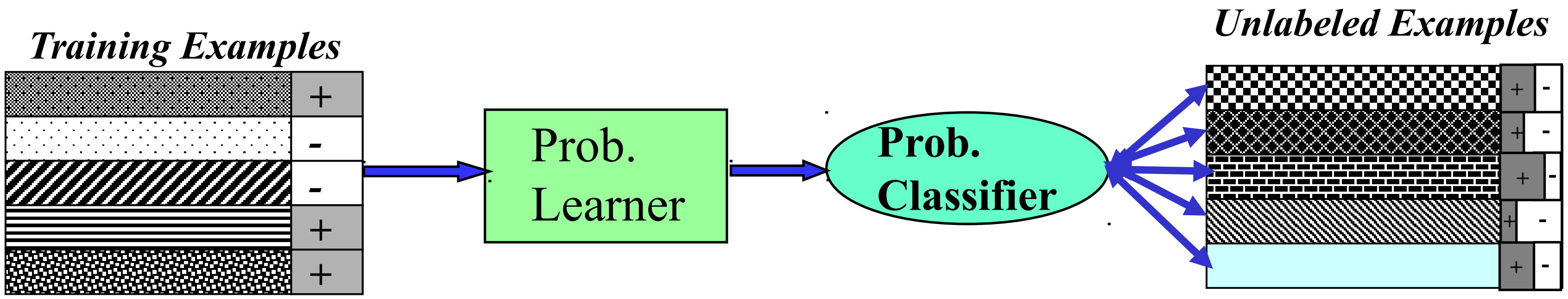
EM Properties

- 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.

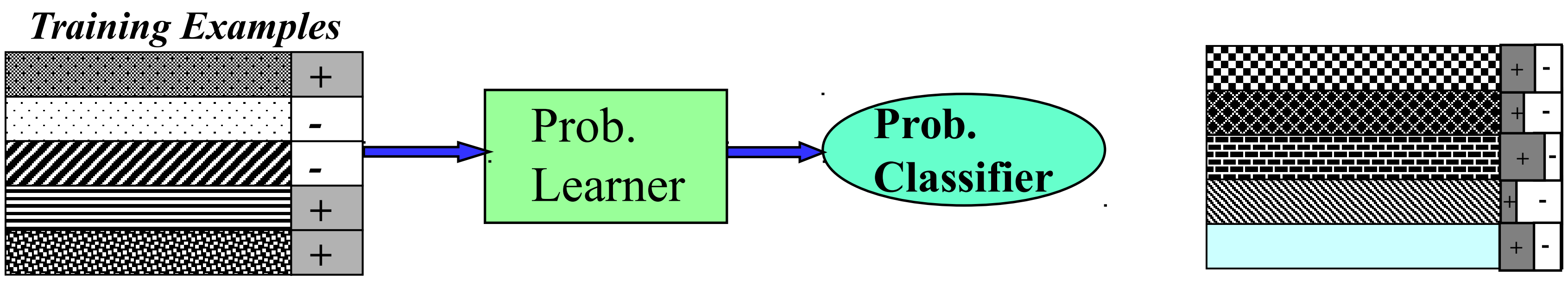
Semi-Supervised Learning

- 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.

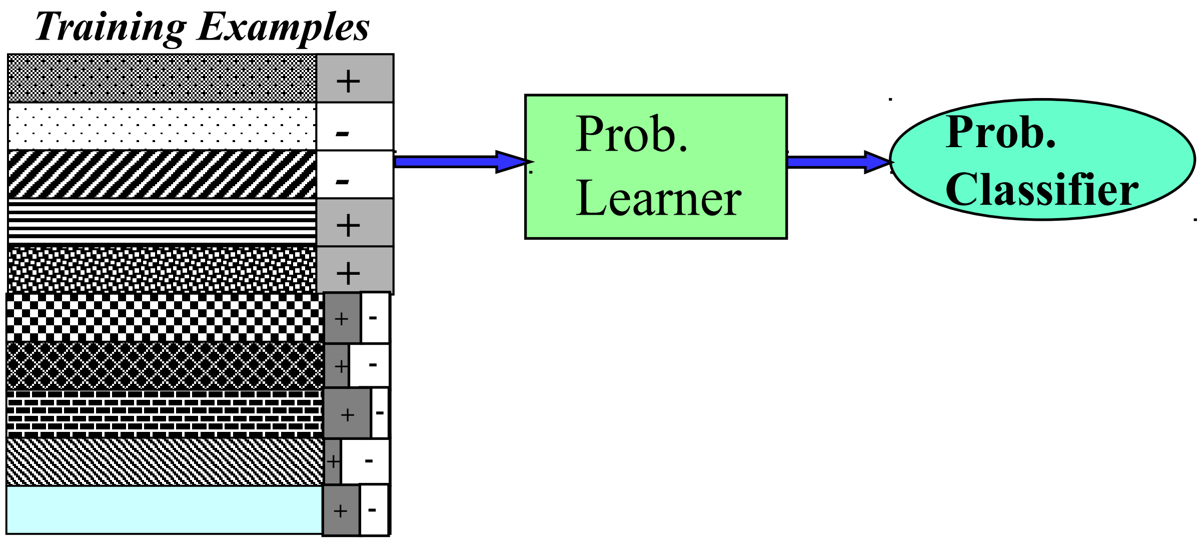
Semi-Supervised EM



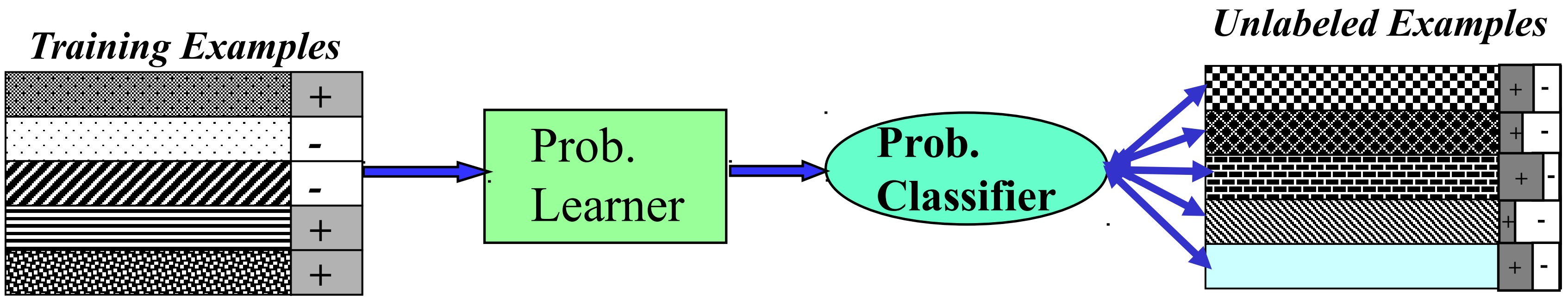
Semi-Supervised EM



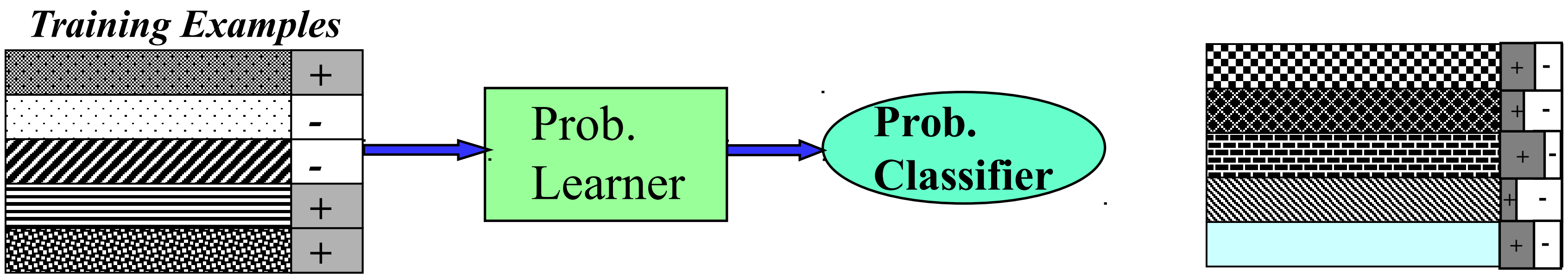
Semi-Supervised EM



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 adalah tingkat terendah pada analisis sintaksis.
- Berupa ihwal pelabelan rangkaian, yaitu sebuah klasifikasi kolektif yang dapat diterapkan di information extraction, phrase chunking, semantic role labeling, and bioinformatics.
- HMM adalah standard generative probabilistic model untuk pelabelan rangkaian yang mengkomputasikan rangkaian paling mungkin dari label secara efisien dan mensupport supervised, unsupervised, dan semi-supervised learning.