BBM 495
Hidden Markov Models (HMMs) and Part-of-Speech Tagging

LECTURER: BURCU CAN

2019-2020 SPRING

Today's topics

- Word Tokenization
- Stemming
- Spelling correction
- PoS Tagging

How many words?

- My cat in the hat is different from other cats!
  - Lemma: same stem
    - cat and cats = same lemma
  - Word form: the inflected surface form
    - cat and cats = different word forms

How many words

- they lay back on the San Francisco grass and looked at the stars and their

- Type: an element of the vocabulary
- Token: an instance of that type

- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

Issues in tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

Tokenization: language issues

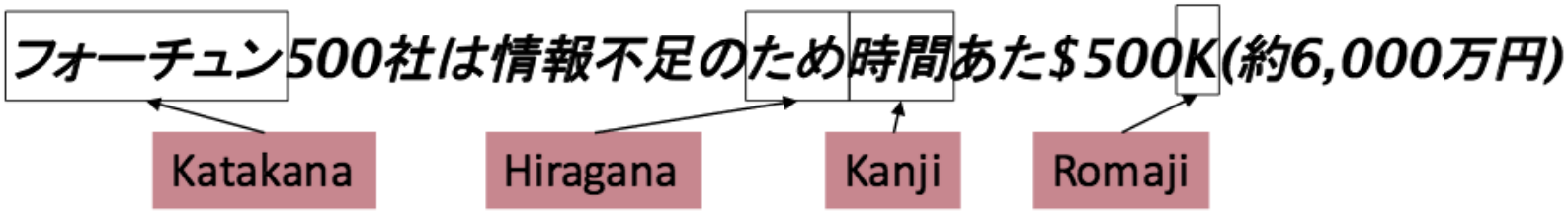
- French
  - L'ensemble → one token or two?
    - L? L'? Le?
    - Want l'ensemble to match with un ensemble

- German noun compounds are not segmented
  - Lebensversicherungsgesellschaftsangestellter
    - 'life insurance company employee'
  - German information retrieval needs compound splitter

Tokenization: language issues

- Chinese and Japanese no spaces between words
- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Sharapova now lives in US southeastern Florida

- Further complicated in Japanese, with multiple alphabets intermingled



Word tokenization in Chinese

- Also called Word Segmentation
- Chinese words are composed of characters.
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum matching (also called ‘greedy’)

Maximum matching word segmentation algorithm

- Given a wordlist of Chinese, and a string
  1. Start a pointer at the beginning of the string
  2. Find the longest word in dictionary that matches the string starting at pointer
  3. Move the pointer over the word in string
  4. Go to 2

An illustration

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there
- — theta bled own there

- Doesn’t generally work in English!

- But works astonishingly well in Chinese

- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

Stemming

Stemming

- Reduce terms to their stems in information retrieval
- Stemming is crude chopping of affixes.
  - Language dependent
  - e.g. **automate(s), automatic, automation** all reduced to **automat**

Porter's algorithm

the most common English stemmer

Step 1a

sses	→ ss	caresses	→ caress
ies	→ i	ponies	→ poni
ss	→ ss	caress	→ caress
s	→ ø	cats	→ cat

Step 1b

(*v*)ing	→ ø	walking	→ walk
		sing	→ sing
(*v*)ed	→ ø	plastered	→ plaster
…			

Step 2 (for long stems)

ational	→ ate	relational	→ relate
izer	→ ize	digitizer	→ digitize
ator	→ ate	operator	→ operate
…			

Step 3 (for longer stems)

al	→ ø	revival	→ reviv
able	→ ø	adjustable	→ adjust
ate	→ ø	activate	→ activ
…			

Spell checking

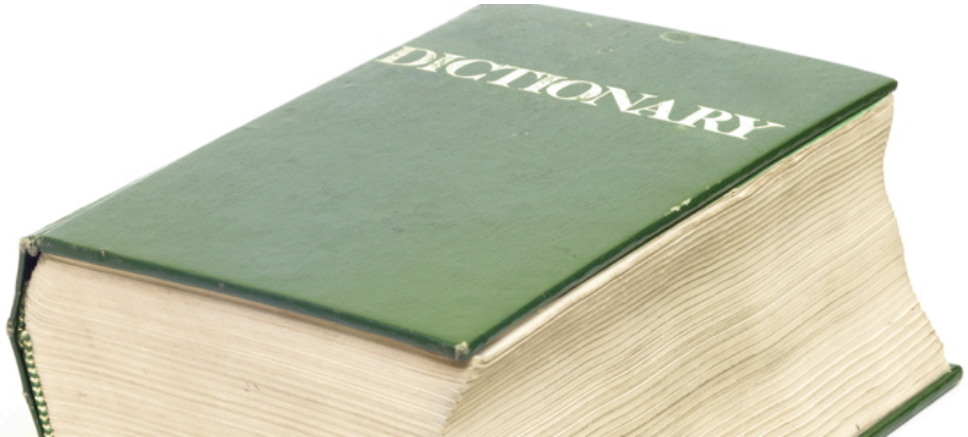
Spell checking and edit distance

- Non-word error detection
  - Detecting “graffe”
  - Figuring out that “graffe” should be “giraffe”

- Context dependent error detection and correction
  - Figuring out that “war and piece” should be “peace”

Non-word error detection

- Any word not in a dictionary
- Assume it's a spelling error
- Need a big dictionary!
- What to use?



Isolated word error correction

- How do I fix “graffe”?
  - Search through all words:
    - graf
    - craft
    - grail
    - giraffe
  - Pick the one that’s closest to graffe

Edit distance

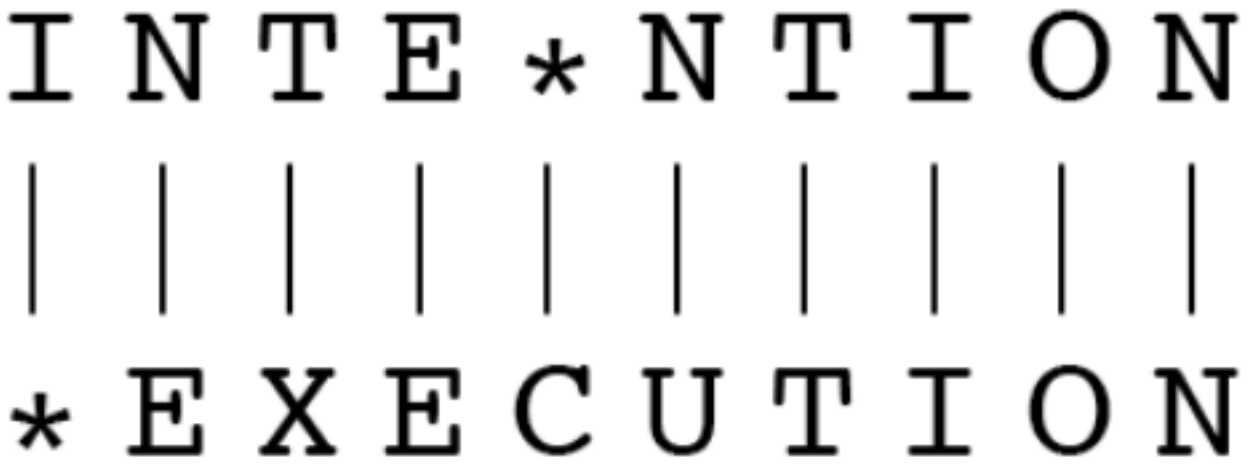
Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	-	deletion
acress	cress	-	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	-	s	insertion
acress	acres	-	s	insertion

Edit distance

- The minimum edit distance between two strings is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution
- Needed to transform one into the other.

Minimum edit distance

- Two strings and their alignment



Minimum edit distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- If each operation has cost 1
  - Distance between these is 5
- If substitution cost 2
  - Distance between them is 8

Minimum edit as search

- Lots of distinct paths during the search
- We don’t have to keep track of all of them
- Just the shortest path to each of those visited states

Minimum edit distance

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Minimum edit distance

N	9									
O	8									
I	7									

$$distance[i,j] = \min \begin{cases} distance[i-1,j] + ins\text{-}cost(target_{i-1}) \\ distance[i-1,j-1] + subst\text{-}cost(source_{j-1}, target \\ distance[i,j-1] + del\text{-}cost(source_{j-1})) \end{cases}$$

T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Minimum edit distance

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

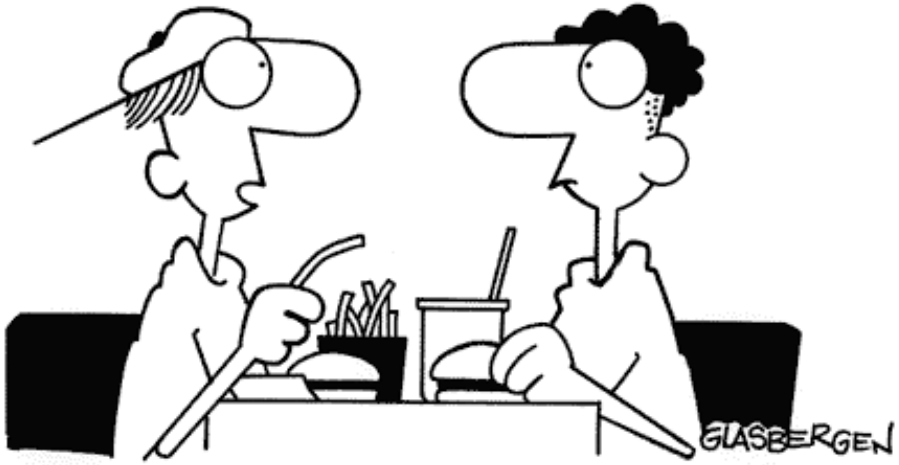
Alignment

- Suppose we want the alignment too:
  - We can keep a “backtrace”
  - Every time we enter a cell, remember where we came from
  - Then when we reach the end, we can trace back from the upper right corner to get an alignment

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ **8**	
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ **8**	← 9	
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ **8**	← 9	← 10	
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ **8**	← 9	← 10	←↓ 11	
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ **8**	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10	
e	4	↙ 3	← 4	↙← **5**	← **6**	← 7	←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	
t	3	↙←↓ 4	↙←↓ **5**	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙ 7	←↓ 8	↙←↓ 9	↓ 8	
n	2	↙←↓ **3**	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↓ 7	↙←↓ 8	↙ 7	
i	**1**	↙←↓ 2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙ 6	← 7	← 8	
#	**0**	1	2	3	4	5	6	7	8	9	
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**	

Copyright 1997 Randy Glasbergen. www.glasbergen.com



"I forgot to make a back-up copy of my brain, so everything I learned last semester was lost."

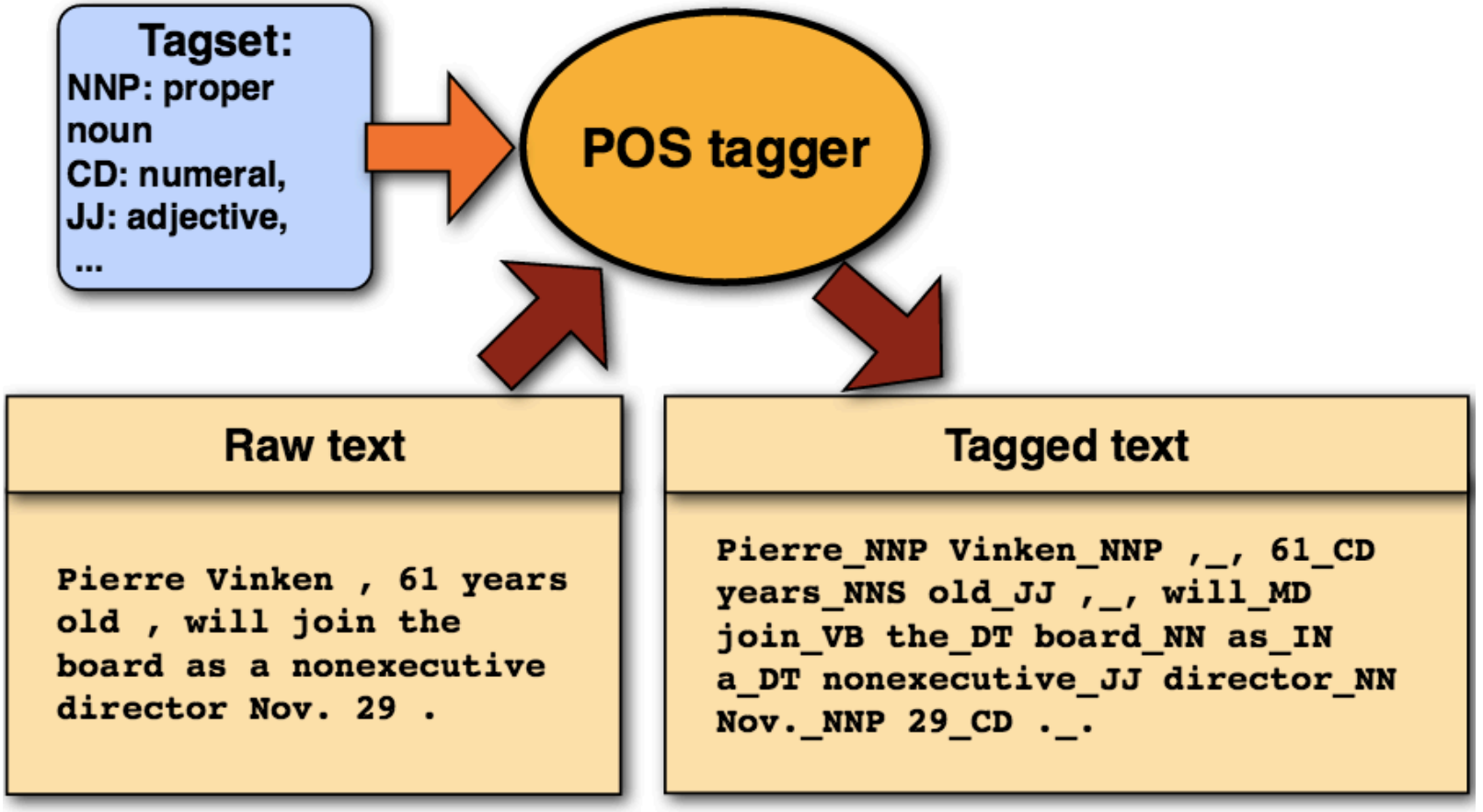
Remember later: We will see a probabilistic version of this called "Viterbi"!

POS Tagging

Frank and Ernest

© by Thaves. Distributed from www.thecomics.com.

Part-of-Speech tags



Part-of-Speech (PoS)

- Each word belongs to a word class. The word class of a word is known as part-of-speech (POS) of that word.

POS tagging

- The POS tagging task is to determine the POS tag of a particular instance of a word.
- Most of words have a single POS tag, but some of them have more than one (2,3,4,…)
- For example, book/noun or book/verb
  - **I bought a *book*.**
  - **Please *book* that flight.**

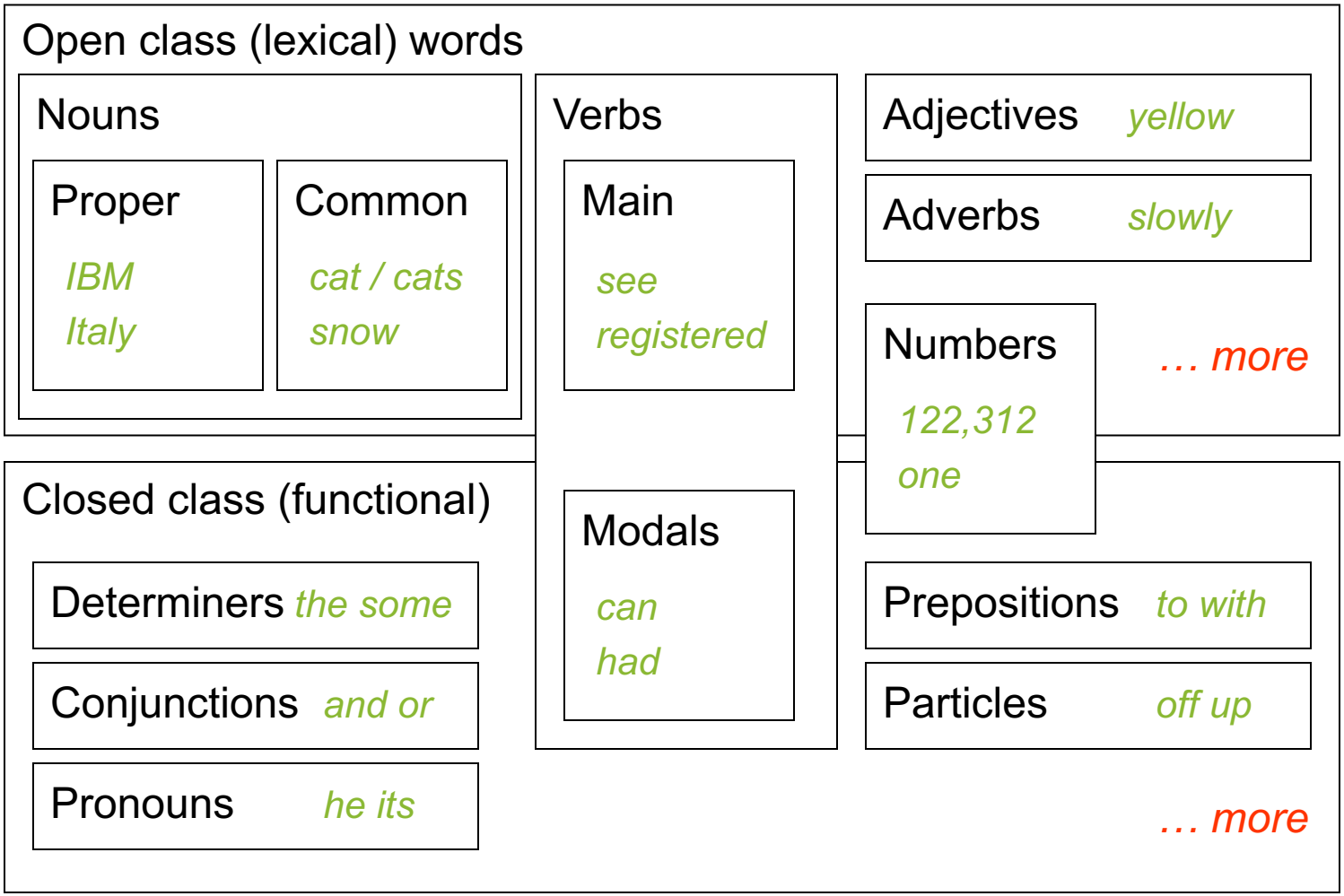
Why PoS tagging?

- Speech synthesis
  - How to pronounce “lead” in English?
  - How to pronounce “koyun” in Turkish?
  - INsult or inSULT, OBject or obJECT, OVERflow or overFLOW
- Information extraction
  - Finding names, relations, etc.
- Machine translation
  - The noun “content’ may have different translations.

Tag sets

- There are various tag sets to choose.
- The choice of the tag set depends on the nature of the application.
  - We may use small tag set (more general tags) or
  - large tag set (finer tags).
- Some of widely used part-of-speech tag sets:
  - Penn Treebank has 45 tags
  - Brown Corpus has 87 tags
  - C7 tag set has 146 tags
- In a tagged corpus, each word is associated with a tag from the used tag set.

Parts-of-Speech (English)



English word classes

- Part-of-speech can be divided into two broad categories:
  - **closed class types** -- such as prepositions
  - **open class types** -- such as noun, verb

- Closed class words are generally also **function words**.
  - Function words play important role in grammar
  - Some function words are: *of, it, and, you*
  - Functions words are most of time very short and frequently occur.

- There are four major open classes.
  - noun, verb, adjective, adverb
  - a new word may easily enter into an open class.
- Word classes may change depending on the natural language, but all natural languages have at least two word classes: *noun* and *verb*.

English PoS Tags

CC	conjunction, coordinating	and both but either or
CD	numeral, cardinal	mid-1890 nine-thirty 0.5 one
DT	determiner	a all an every no that the
EX	existential there	there
FW	foreign word	gemeinschaft hund ich jeux
IN	preposition or conjunction, subordinating	among whether out on by if
JJ	adjective or numeral, ordinal	third ill-mannered regrettable
JJR	adjective, comparative	braver cheaper taller
JJS	adjective, superlative	bravest cheapest tallest
MD	modal auxiliary	can may might will would
NN	noun, common, singular or mass	cabbage thermostat investment subhumanity
NNP	noun, proper, singular	Motown Cougar Yvette Liverpool
NNPS	noun, proper, plural	Americans Materials States
NNS	noun, common, plural	undergraduates bric-a-brac averages
POS	genitive marker	' 's
PRP	pronoun, personal	hers himself it we them
PRP$	pronoun, possessive	her his mine my our ours their thy your
RB	adverb	occasionally maddeningly adventurously
RBR	adverb, comparative	further gloomier heavier less-perfectly
RBS	adverb, superlative	best biggest nearest worst
RP	particle	aboard away back by on open through
TO	"to" as preposition or infinitive marker	to
UH	interjection	huh howdy uh whammo shucks heck
VB	verb, base form	ask bring fire see take
VBD	verb, past tense	pleaded swiped registered saw
VBG	verb, present participle or gerund	stirring focusing approaching erasing
VBN	verb, past participle	dilapidated imitated reunifed unsettled
VBP	verb, present tense, not 3rd person singular	twist appear comprise mold postpone
VBZ	verb, present tense, 3rd person singular	bases reconstructs marks uses
WDT	WH-determiner	that what whatever which whichever
WP	WH-pronoun	that what whatever which who whom
WP$	WH-pronoun, possessive	whose
WRB	Wh-adverb	however whenever where why

TURKISH PoS Tags

Slot Groups	Slot Values
Main POS	Adj, Adv, Conj, Det, Dup, Interj, Noun, Num, Postp, Pron, Punc, Verb
Minor POS	Able, Acquire, ActOf, Adamantly, AfterDoingSo, Agt, Almost, As, AsIf, AsLongAs, Become, ByDoingSo, Card, Caus, DemonsP, Dim, Distrib, EverSince, FeelLike, FitFor, FutPart, Hastily, InBetween, Inf, Inf1, Inf2, Inf3, JustLike, Ly, Ness, NotState, Ord, Pass, PastPart, PCAbl, PCAcc, PCDat, PCGen, PCIns, PCNom, Percent, PersP, PresPart, Prop, Quant, QuesP, Range, Ratio,Real, Recip, ReflexP, Rel, Related, Repeat, Since, SinceDoingSo, Start, Stay, Time, When, While, With, Without, Zero
Person Agreements	A1pl, A1sg, A2pl, A2sg, A3pl, A3sg
Possessive Agreements	P1pl, P1sg, P2pl, P2sg, P3pl, P3sg, Pnon
Case Markers	Abl, Acc, Dat, Equ, Gen, Ins, Loc, Nom
Polarity	Neg, Pos
Tense/Mood	Aor, Desr, Fut, Imp, Neces, Opt, Pres, Prog1, Prog2, Cop, Cond, Past, Narr
Compund Tense	Comp_Cond, Comp_Narr, Comp_Past
Cop	Cop

Disambiguating Main POS tags for Turkish, Ehsani et al. 2012

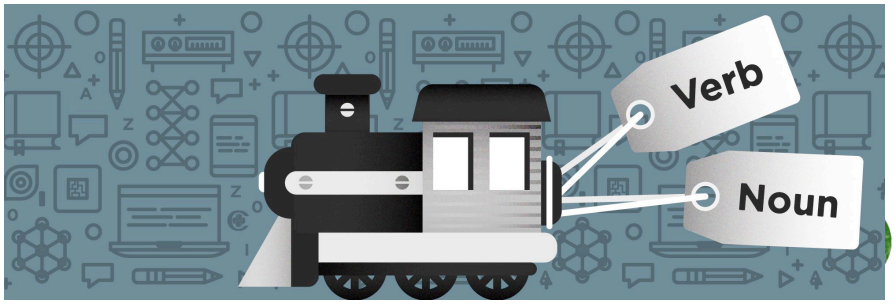
Nouns

- Nouns can be divided as:
  - *proper nouns* -- names for specific entities such as Ankara, John, Ali
  - *common nouns*

Verbs

- Verb class includes the words referring actions and processes.
- Verbs can be divided as:
  - main verbs -- open class -- draw, bake
  - auxiliary verbs -- closed class -- can, should

- Verbs have different morphological forms:
  - non-3rd-person-sg eat
  - 3rd-person-sg - eats
  - progressive -- eating
  - past -- ate
  - past participle -- eaten



Adjectives

- Adjectives describe properties or qualities
  - for color -- black, white
  - for age -- young, old

Adverbs

- Adverbs normally modify verbs.
- Adverb categories:
  - locative adverbs -- home, here, downhill
  - degree adverbs -- very, extremely
  - manner adverbs -- slowly, delicately
  - temporal adverbs -- yesterday, Friday

Prepositions

- Occur before noun phrases indicate spatial or temporal relations
- Example:
  - on the table
  - under chair
- They occur so often. For example, some of the frequency counts in a 16 million word corpora (COBUILD).
  - of 540,085
  - in 331,235
  - for 142,421
  - to 125,691
  - with 124,965
  - on 109,129
  - at 100,169

Articles

- Only three words in the class: a an the

- They occur so often. For example, some of the frequency counts in a 16 million word corpora (COBUILD).
  - the 1,071,676
  - a 413,887
  - an 59,359

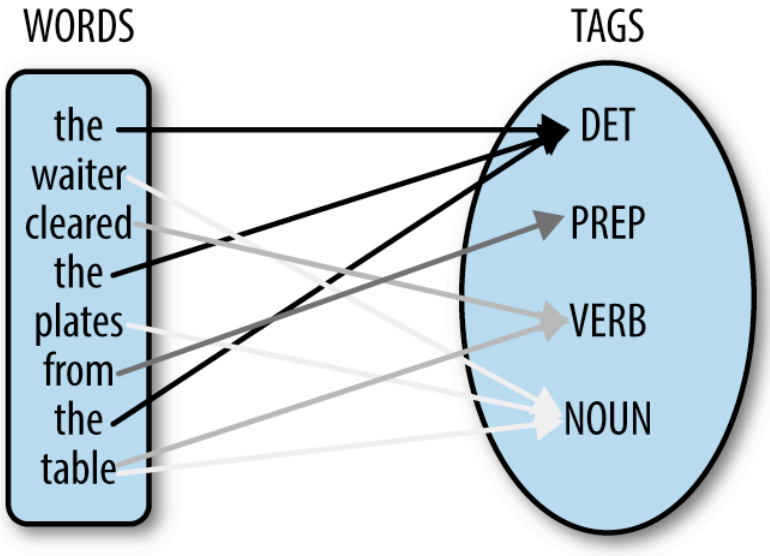
- Almost 10% of words are articles in this corpus.

Pronouns

- Pronouns can be divided:
  - **personal** you she I
  - **possessive** my your his
  - **wh-pronouns** who what -- who is the president?

Building a POS tagger

- Rule-based POS tagging
- Statistical POS tagging

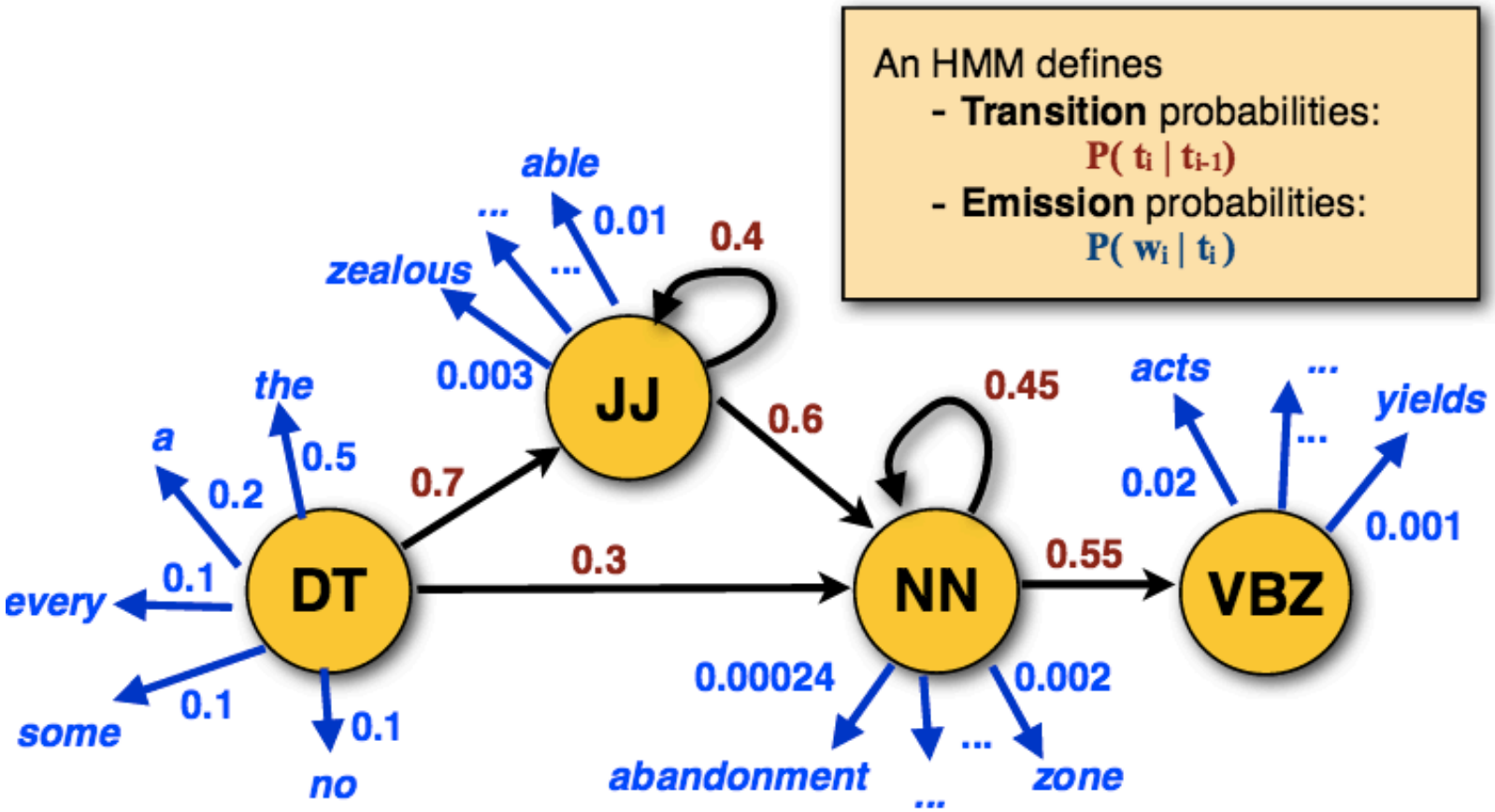


Rule based POS tagging

- Rule-based taggers rely on a dictionary to provide PoS tags for a word, or rules can be learned using training data.

- Example rules:
- If preceding word = ART, then disambiguate {NOUN, VERB} as NOUN

HMM's as probabilistic automata

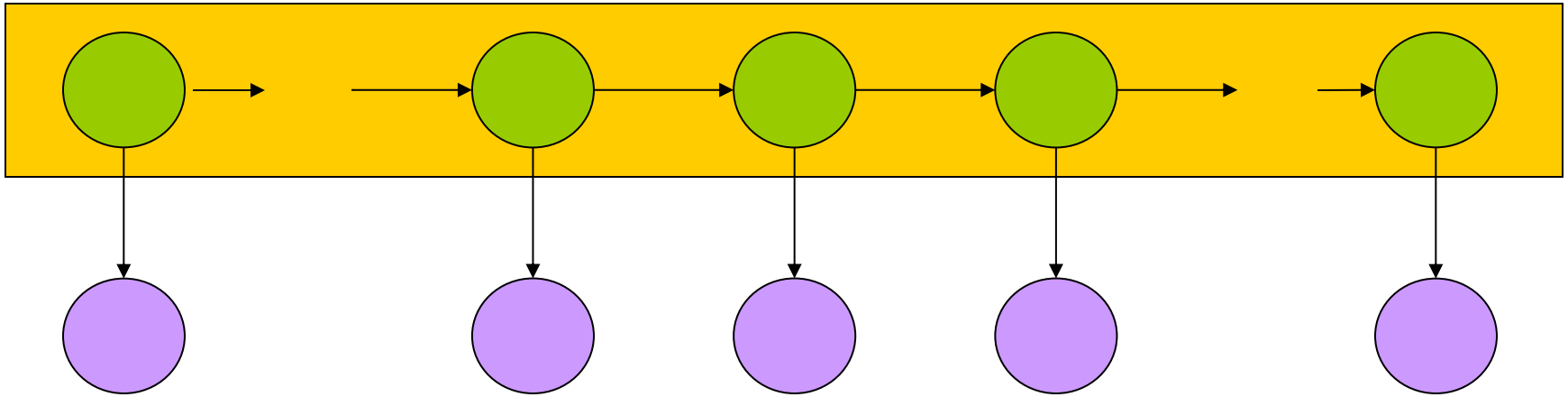


HMM definition

A HMM $\lambda = (A, B, \pi)$ consists of

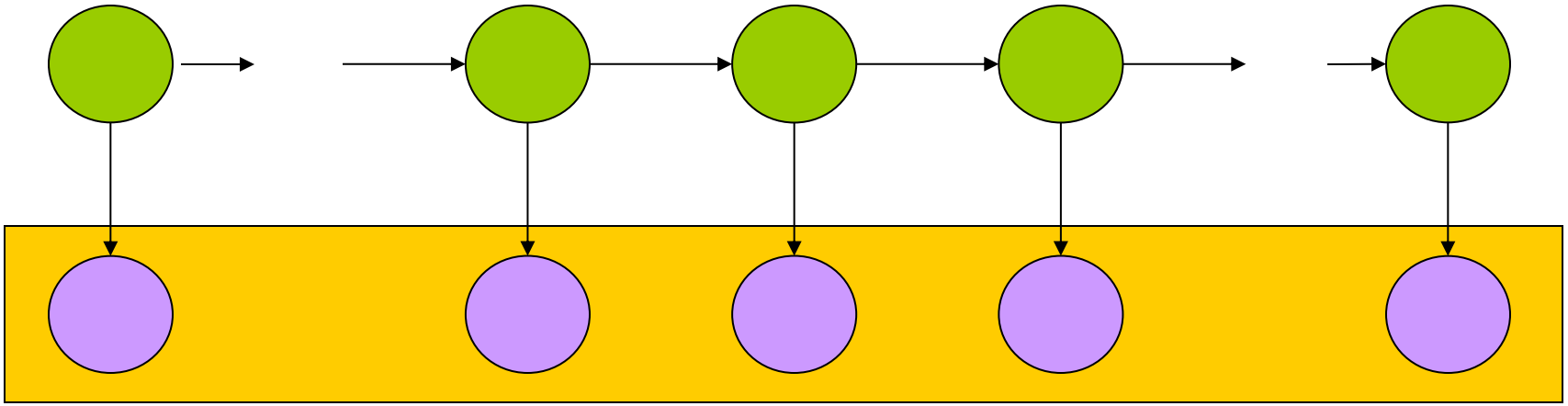
- a set of N **states** $Q = \{q_1, \ldots . q_N\}$
  with $Q_0 \subseteq Q$ a set of **initial states**
  and $Q_F \subseteq Q$ a set of **final (accepting) states**
- an **output vocabulary** of M items $V = \{v_1, \ldots v_m\}$
- an $N \times N$ **state transition probability matrix** A
  with a_{ij} the probability of moving from q_i to q_j.
  $(\sum_{j=1}^{N} a_{ij} = 1 \ \forall i; \quad 0 \leq a_{ij} \leq 1 \ \forall i, j)$
- an $N \times M$ **symbol emission probability matrix** B
  with b_{ij} the probability of emitting symbol v_j in state q_i
  $(\sum_{j=1}^{N} b_{ij} = 1 \ \forall i; \quad 0 \leq b_{ij} \leq 1 \ \forall i, j)$
- an **initial state distribution vector** $\pi = \langle \pi_1, \ldots, \pi_N \rangle$
  with π_i the probability of being in state q_i at time $t = 1$.
  $(\sum_{i=1}^{N} \pi_i = 1 \quad 0 \leq \pi_i \leq 1 \ \forall i)$

What is an HMM?



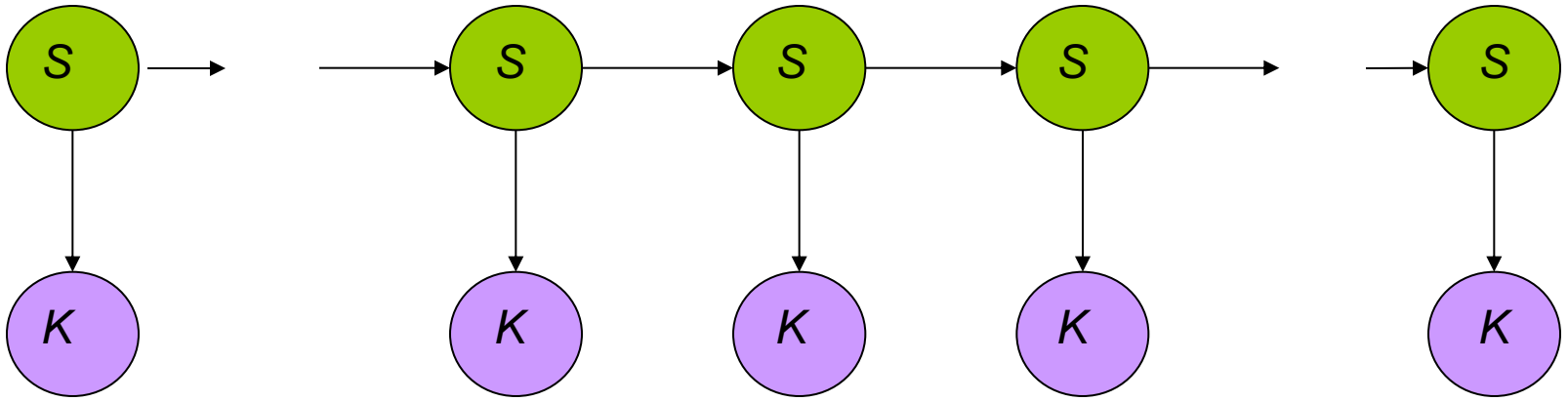
- Green circles are ***hidden states***
- Dependent only on the previous state (bigram)

What is an HMM?



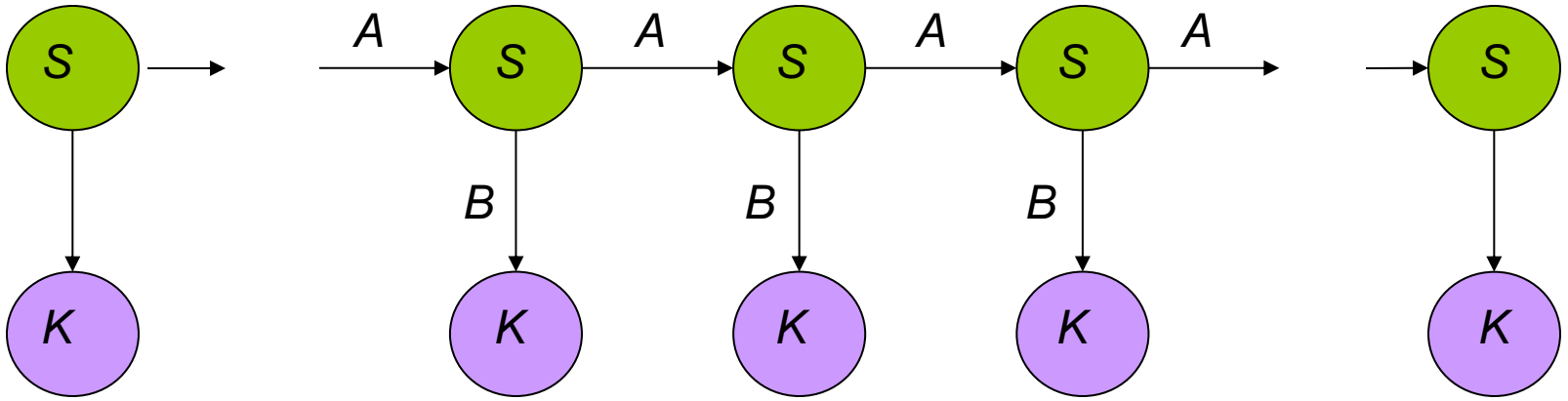
- Purple nodes are ***observed states***
- Dependent only on their corresponding hidden state

HMM Formalism



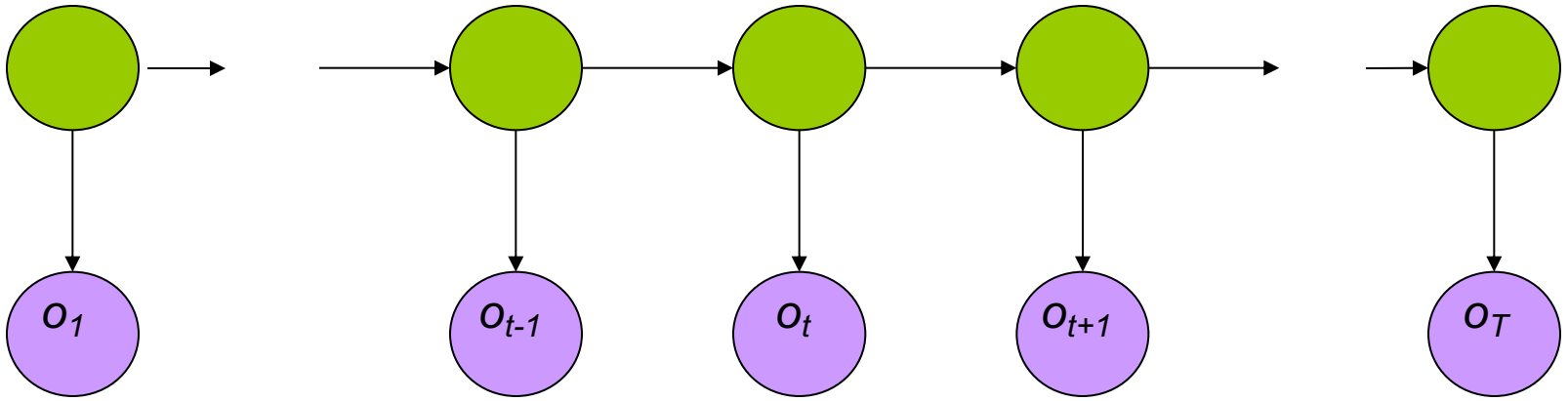
- $\{S, K, \Pi, A, B\}$
- S : $\{s_1 \ldots s_N\}$ are the values for the hidden states
- K : $\{k_1 \ldots k_M\}$ are the values for the observations

HMM Formalism



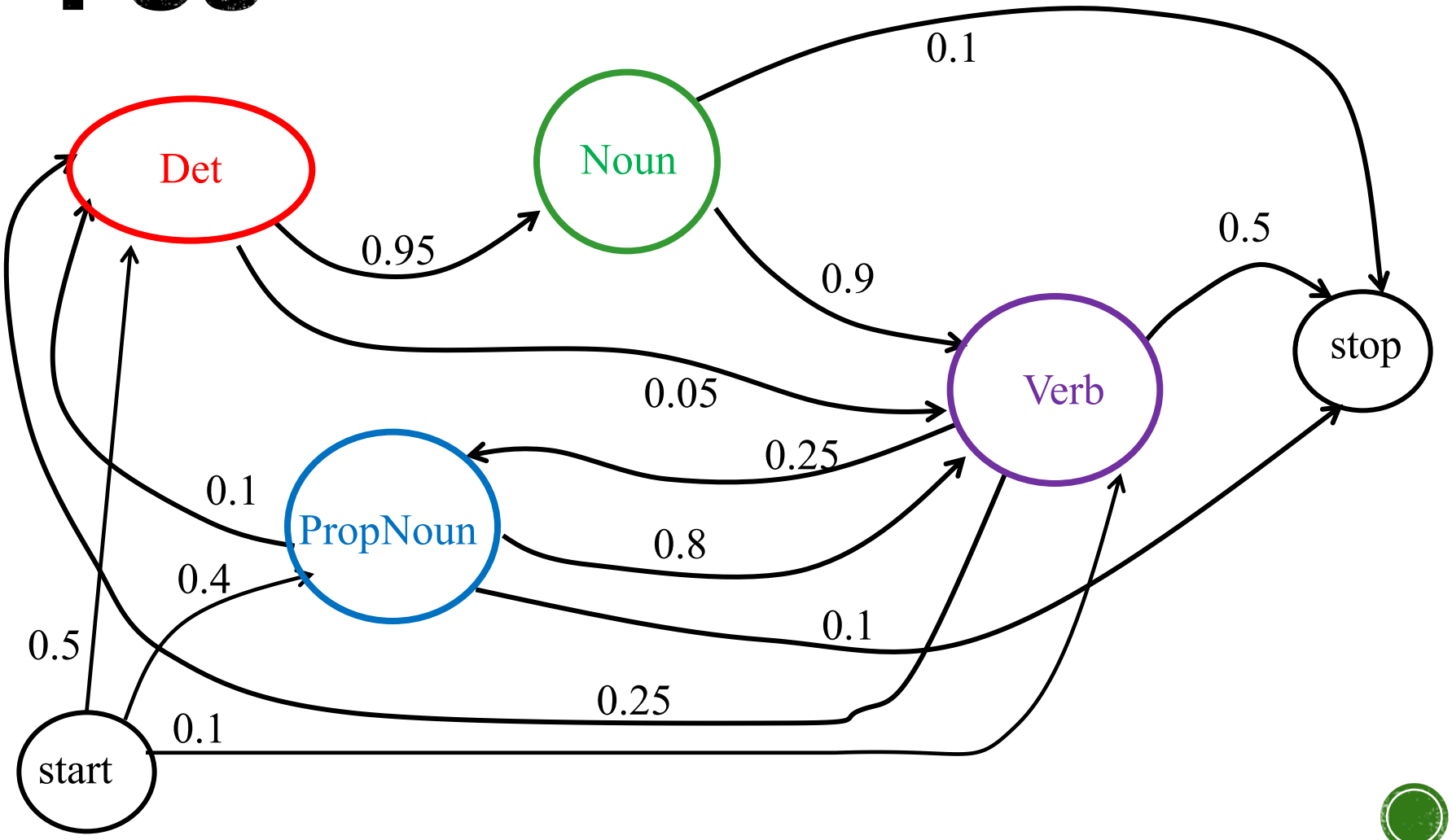
- $\{S, K, \Pi, A, B\}$
- $\Pi = \{\pi_\iota\}$ are the initial state probabilities
- $A = \{a_{ij}\}$ are the state transition probabilities
- $B = \{b_{ik}\}$ are the emission probabilities

Best State Sequence

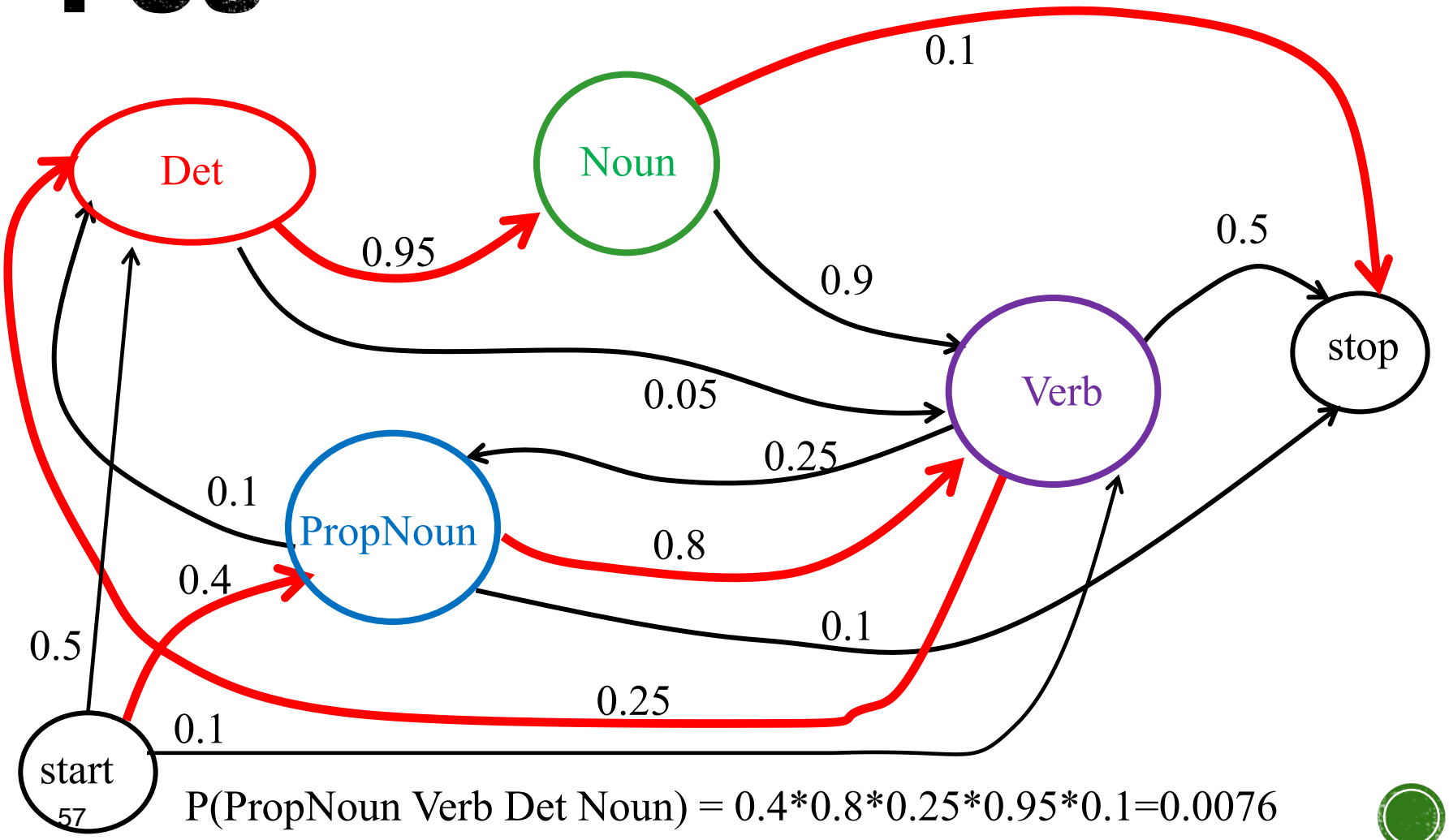


- Find the state sequence that best explains the observations
- **Viterbi** algorithm
- $$\arg\max_X P(X \mid O)$$

Sample Markov Model for POS

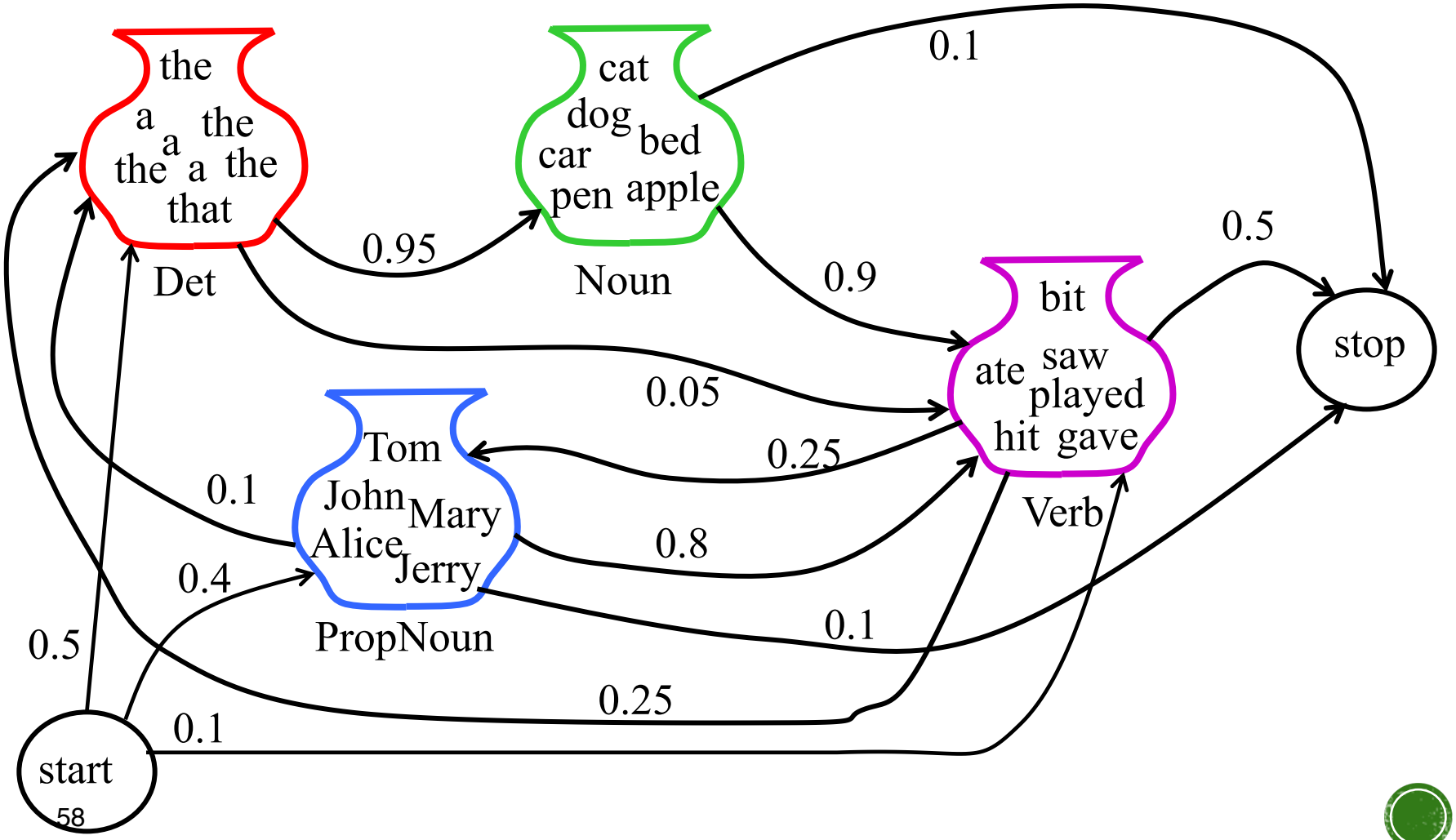


Sample Markov Model for POS

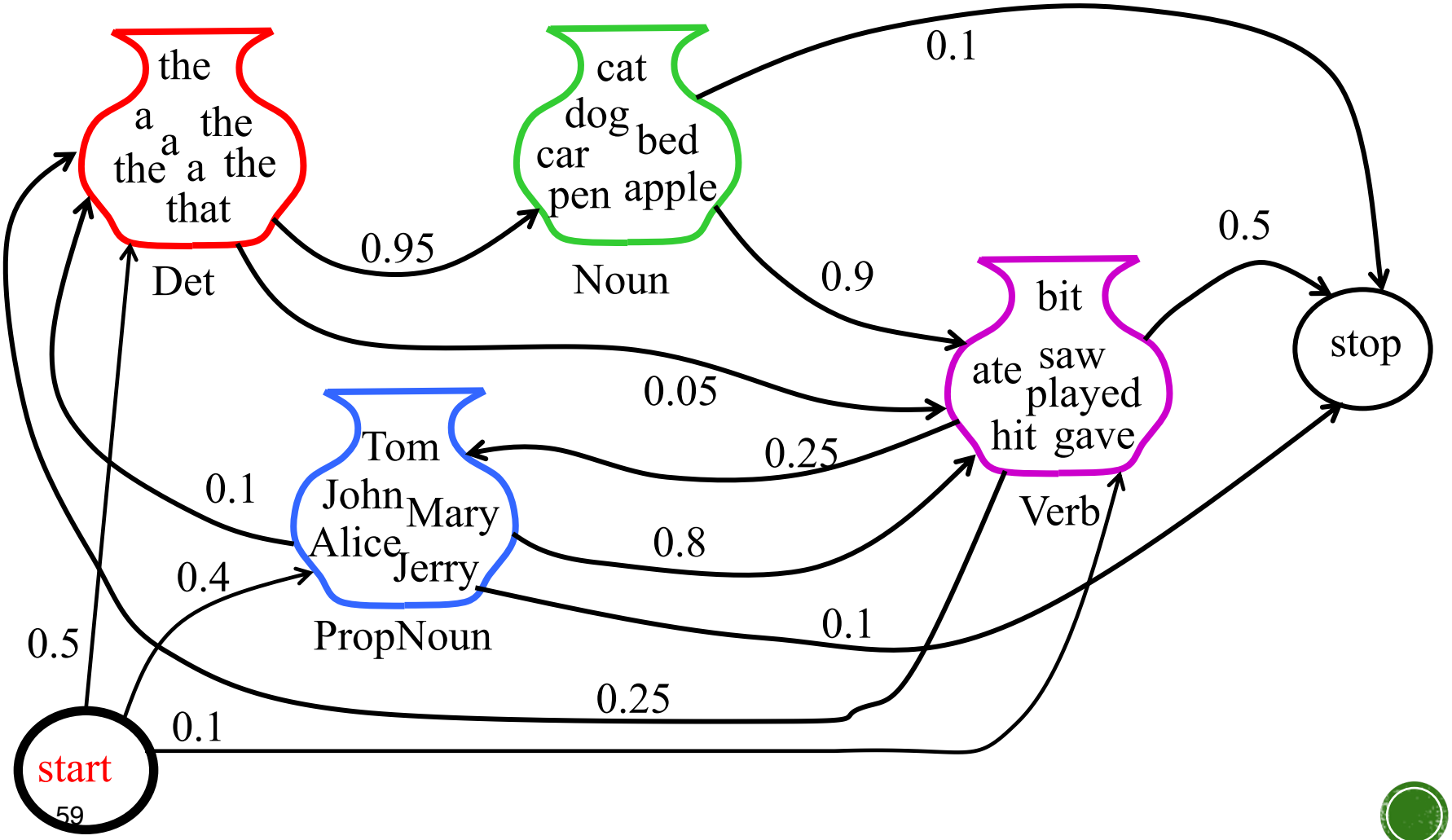


P(PropNoun Verb Det Noun) = 0.4*0.8*0.25*0.95*0.1=0.0076

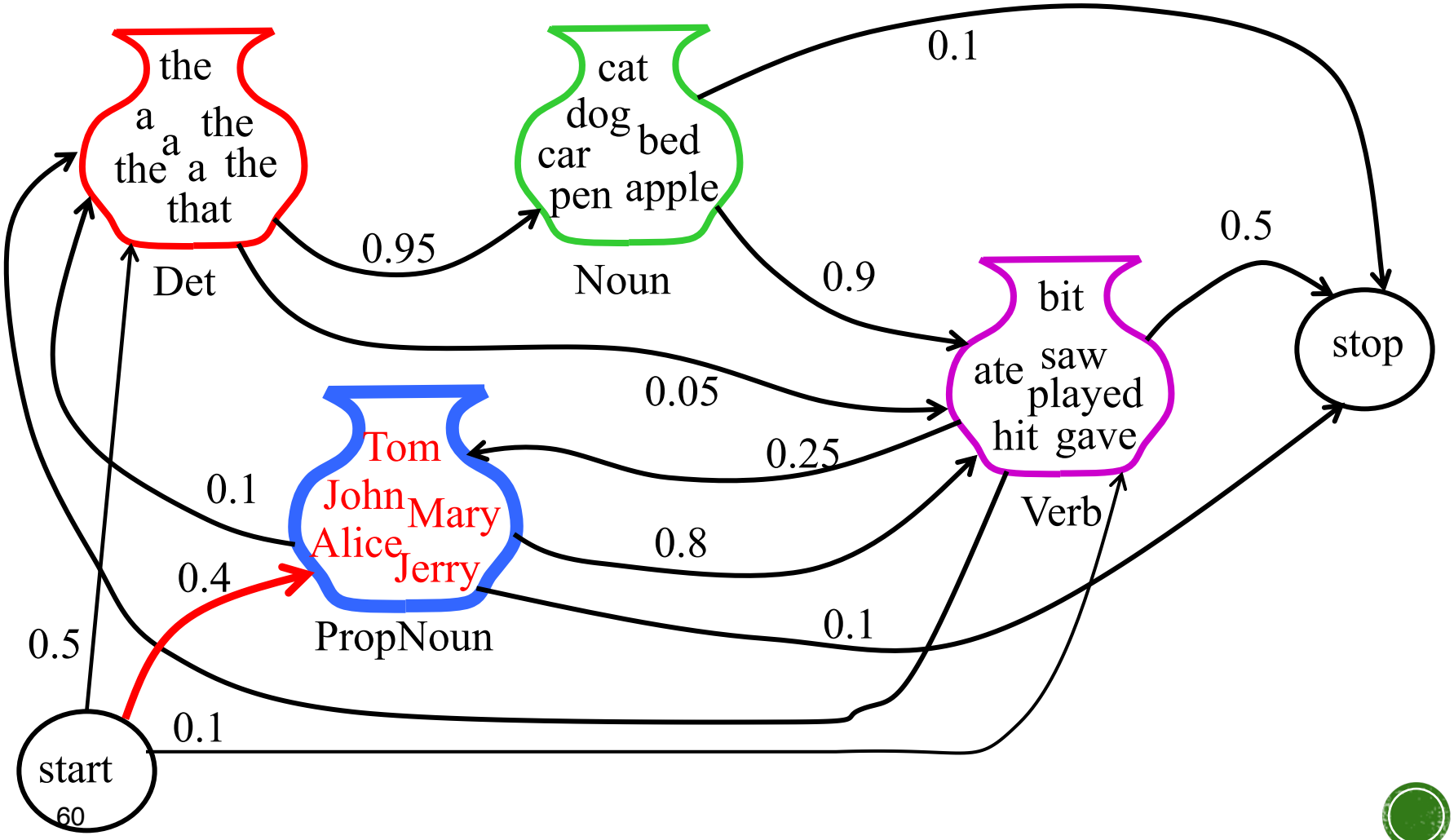
Sample HMM for POS



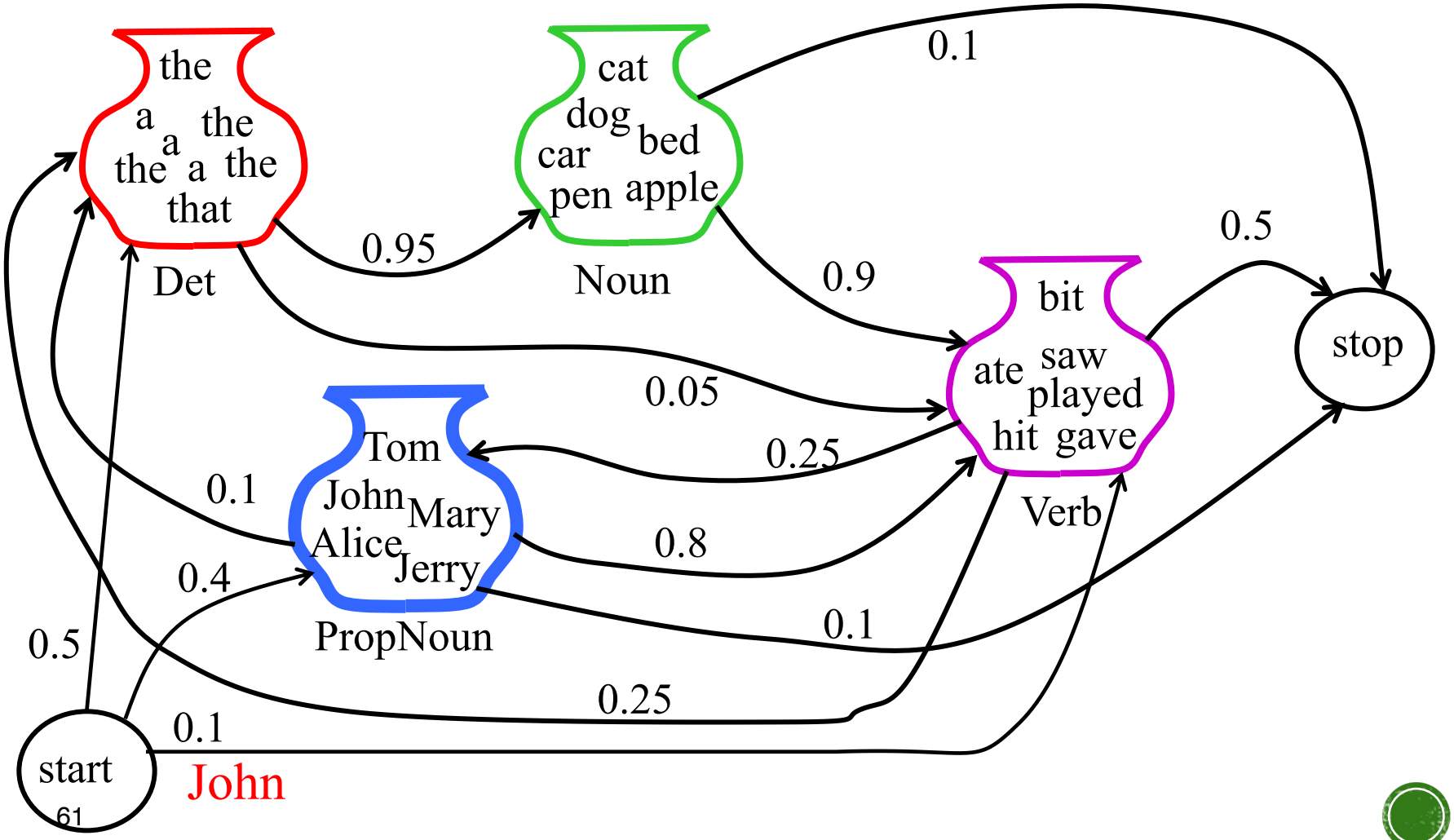
Sample HMM Generation



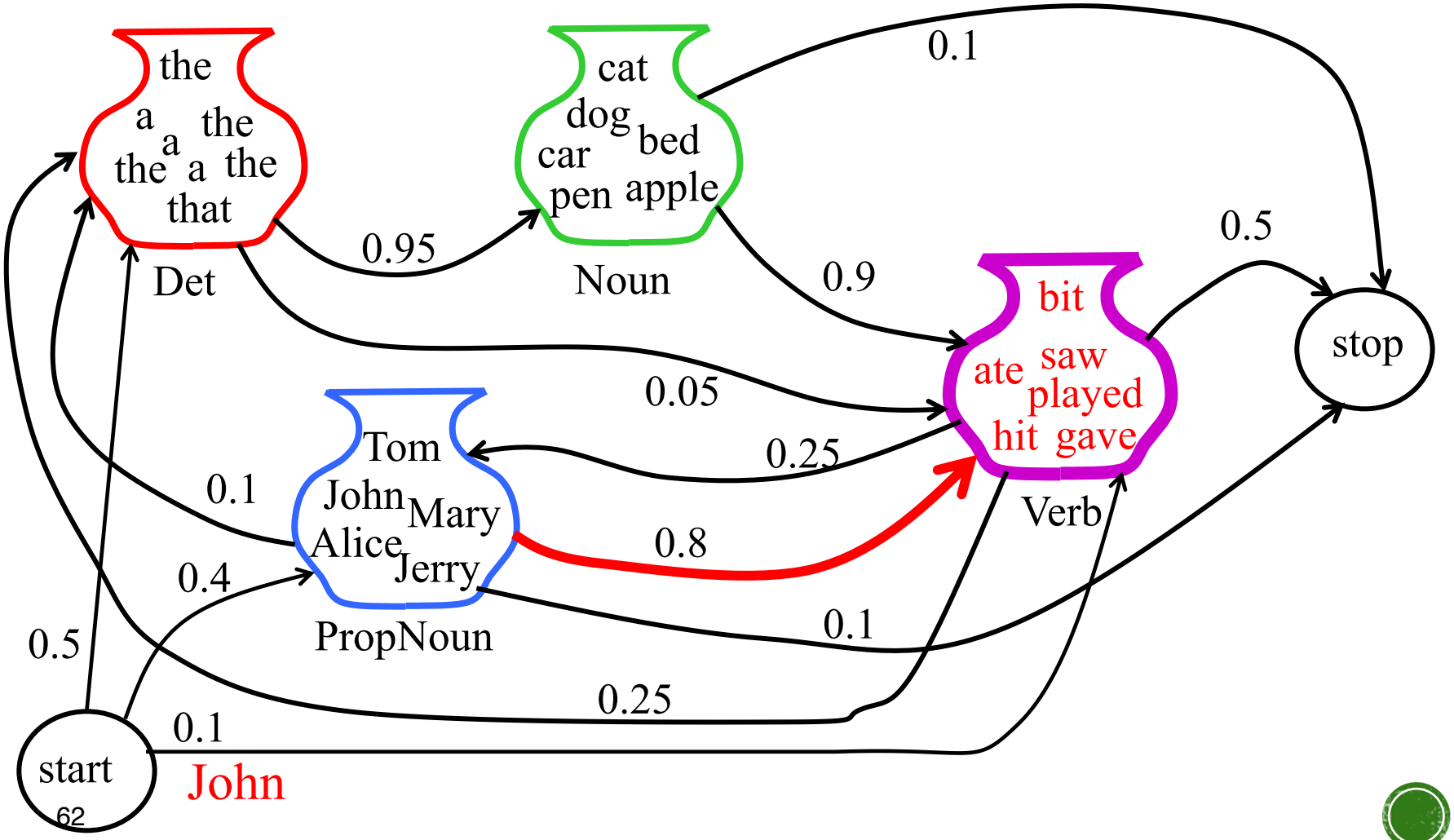
Sample HMM Generation



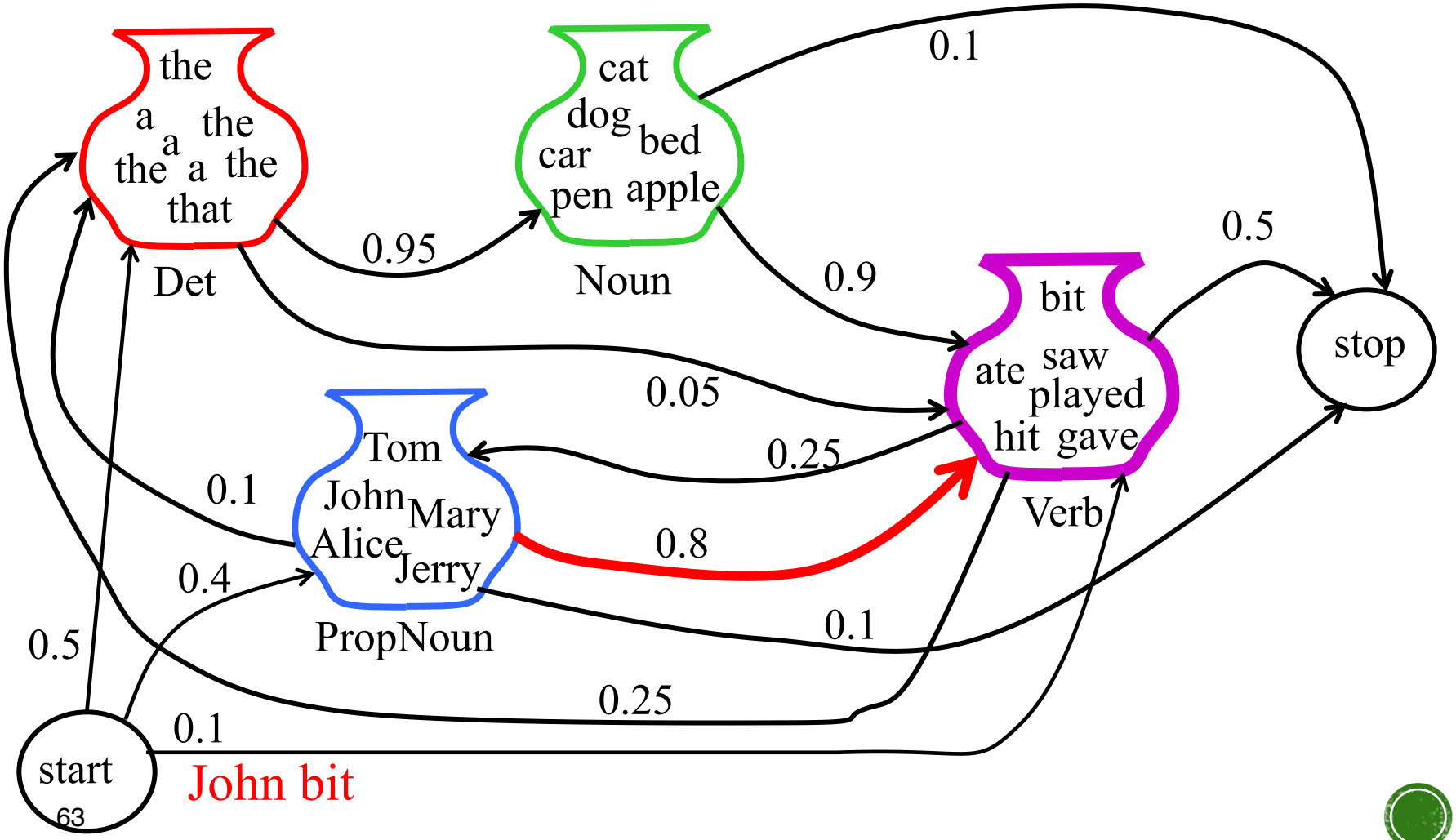
Sample HMM Generation



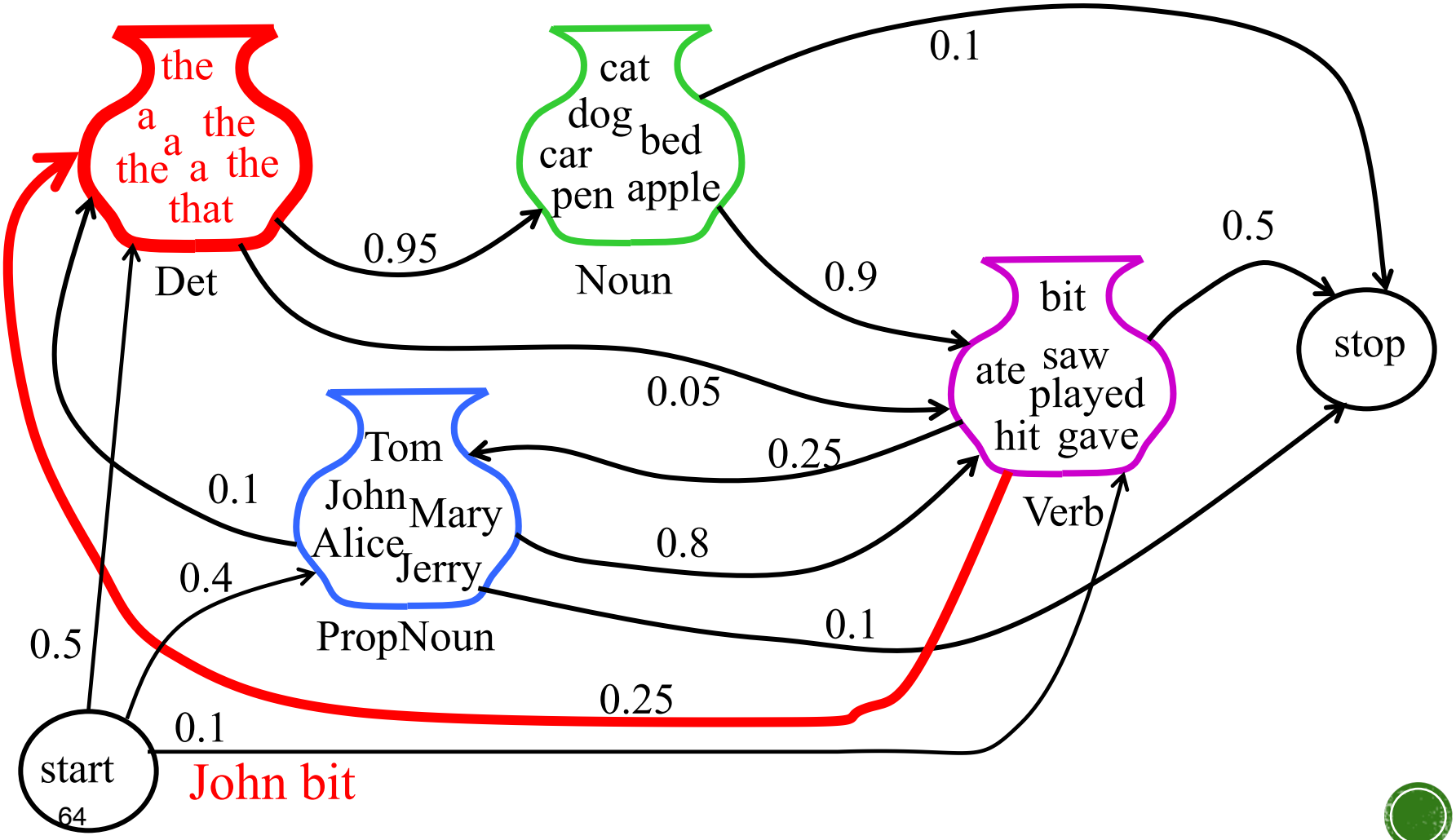
Sample HMM Generation



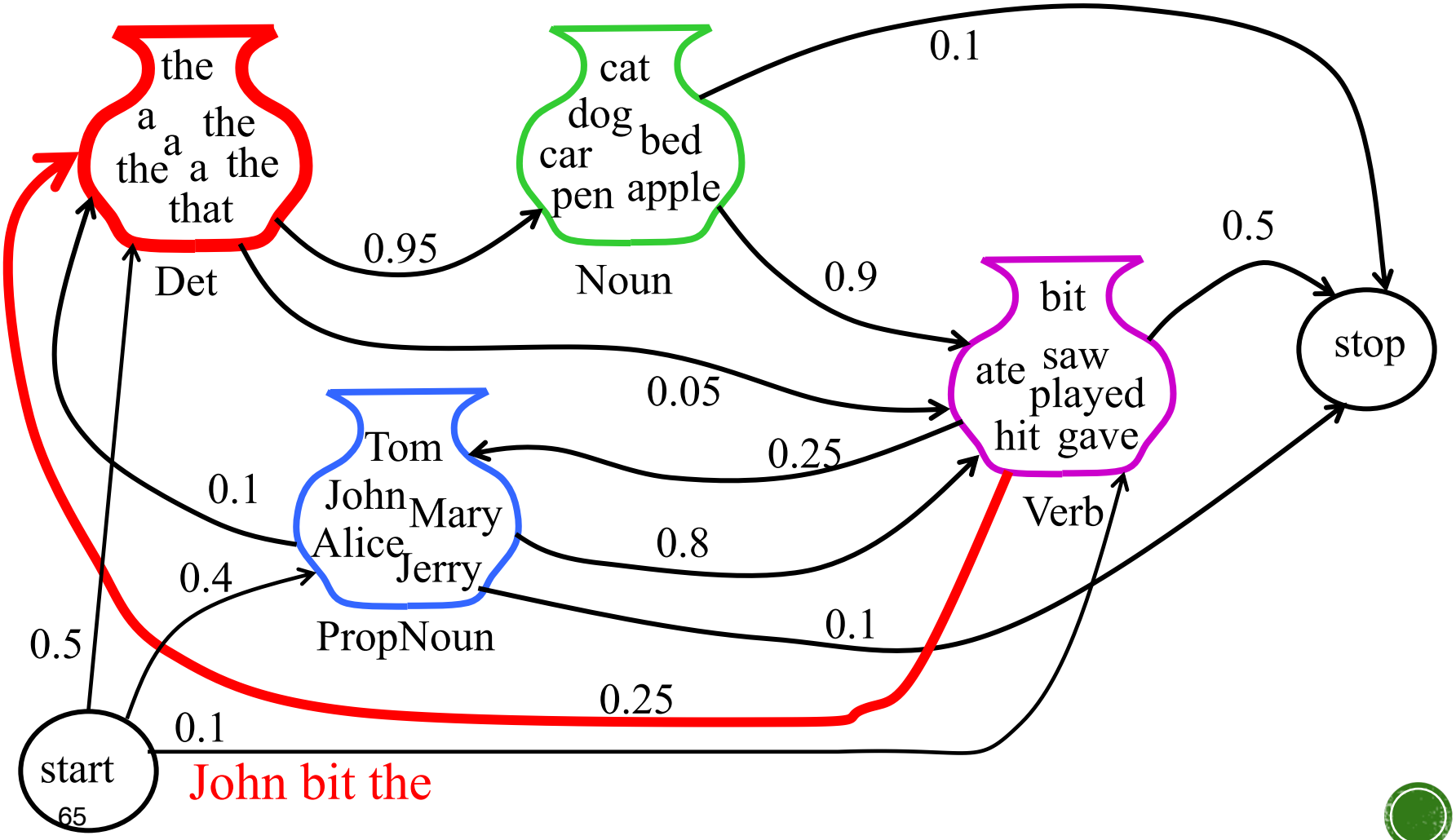
Sample HMM Generation



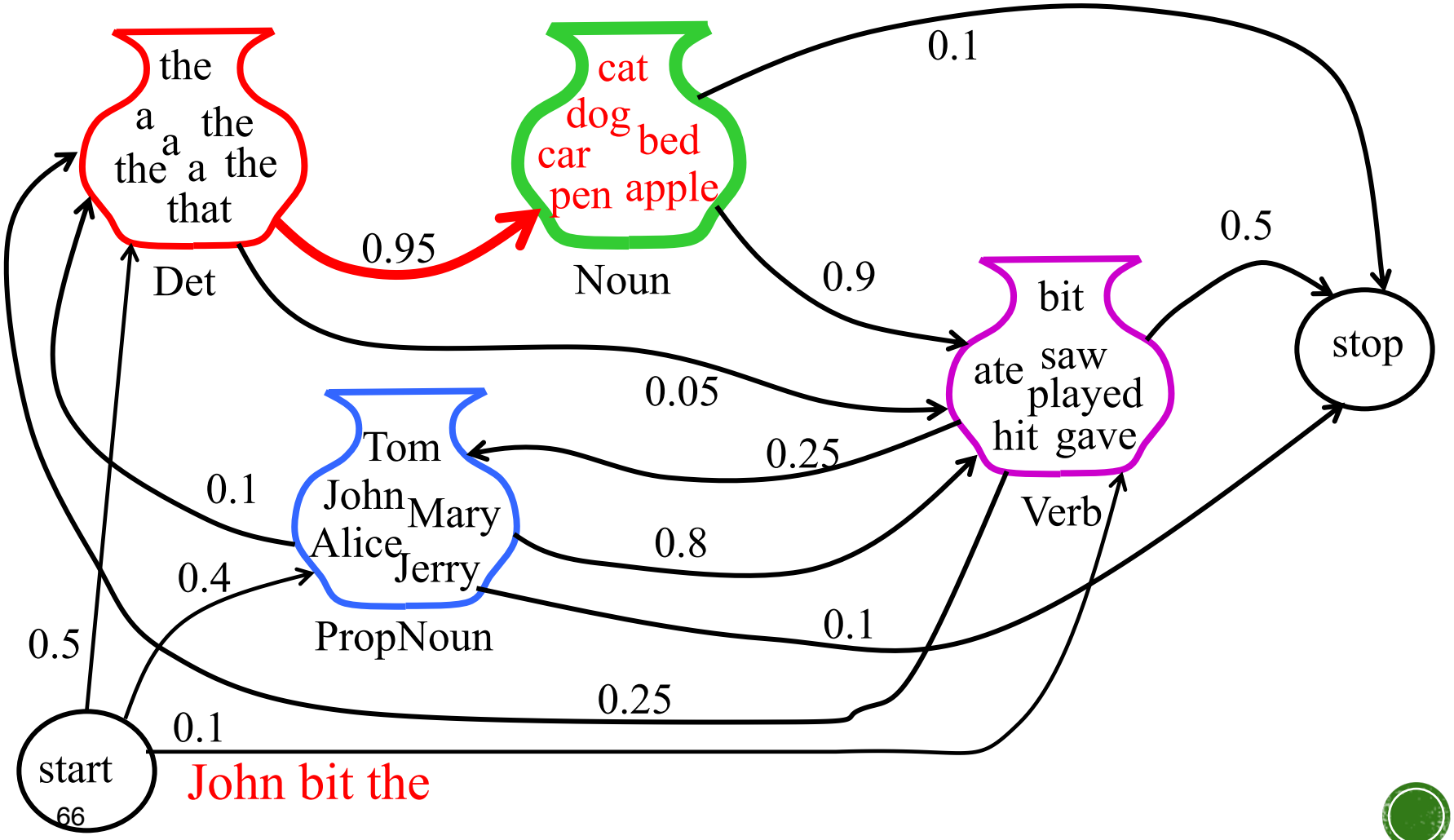
Sample HMM Generation



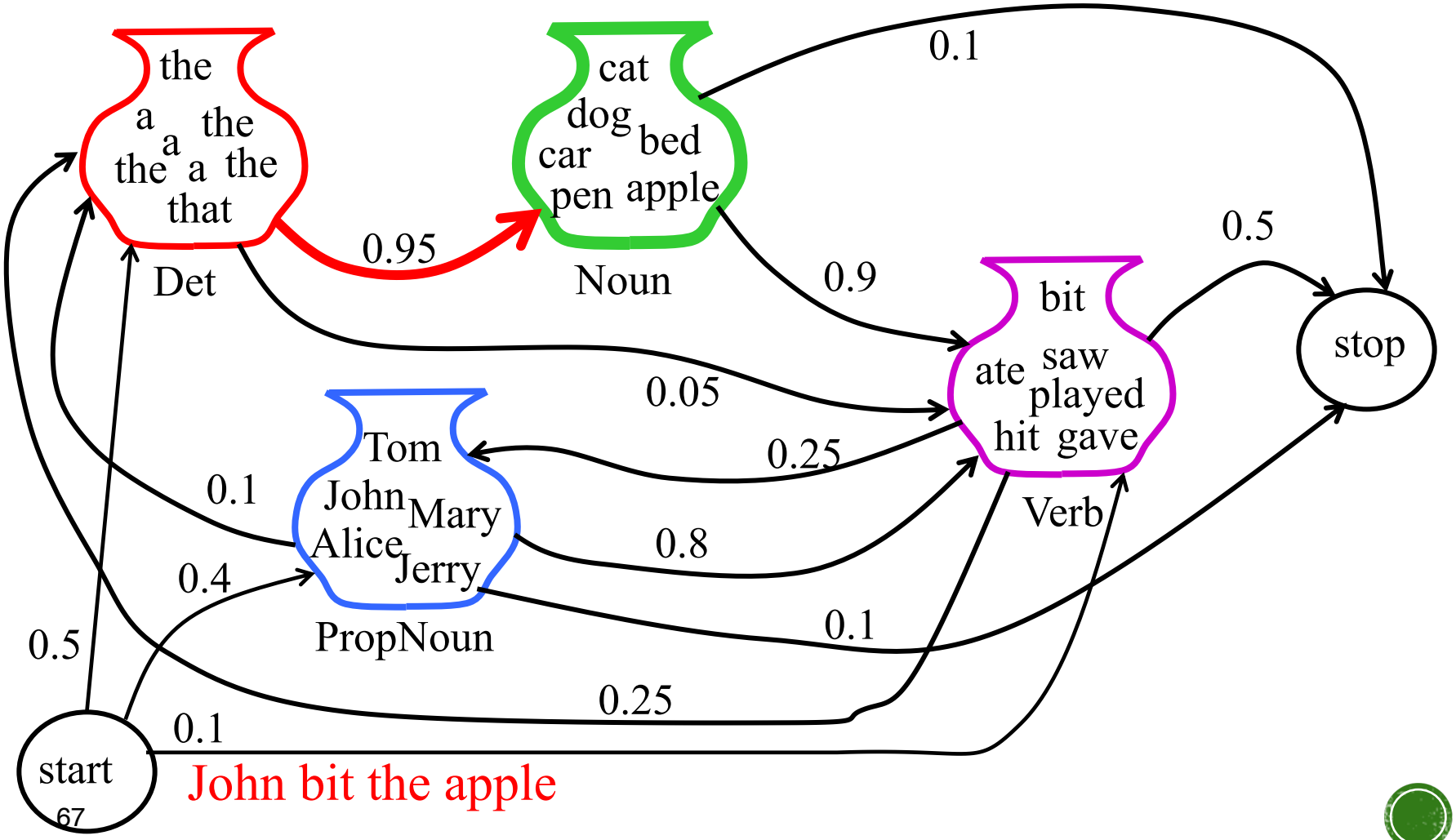
Sample HMM Generation



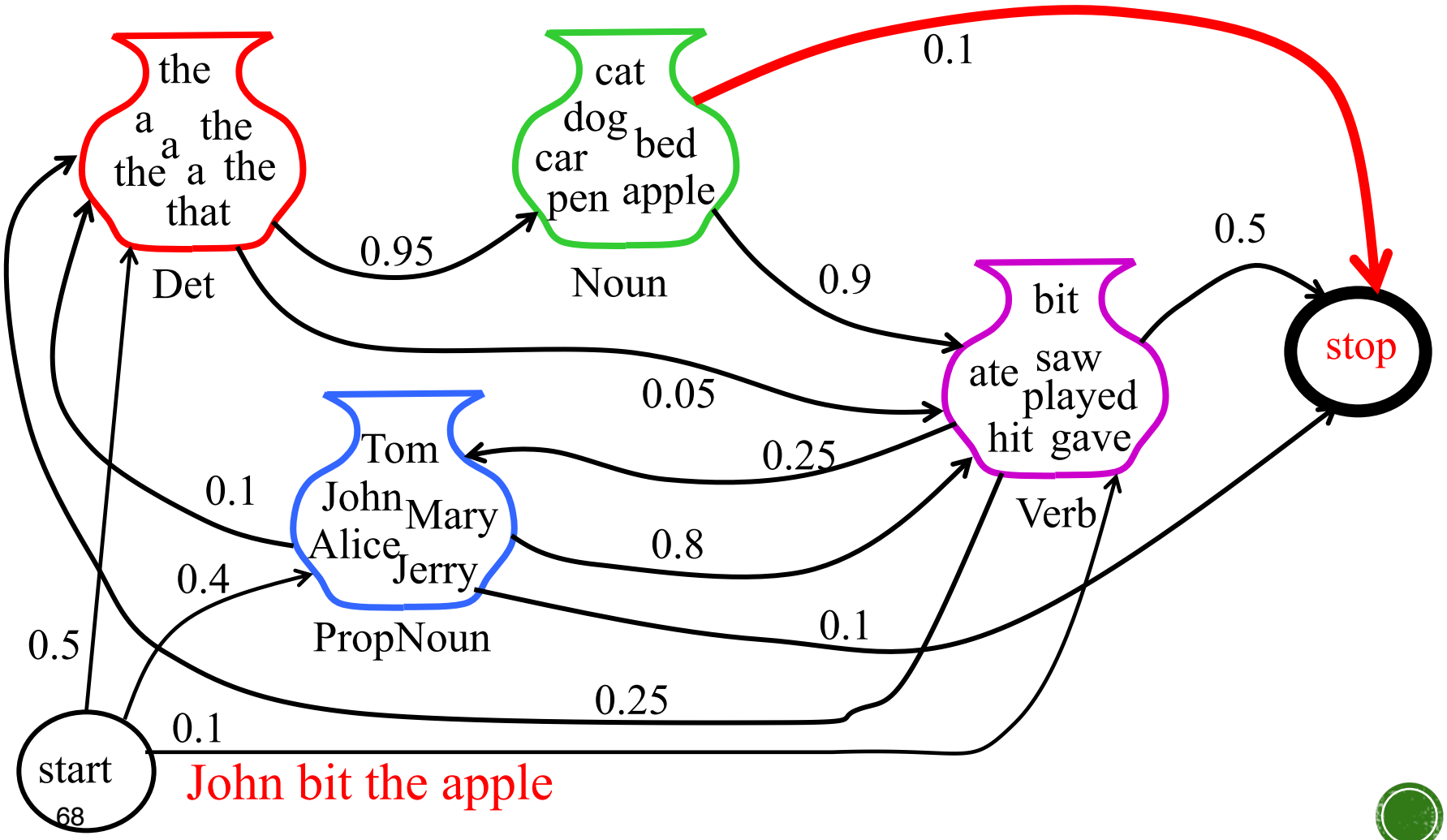
Sample HMM Generation



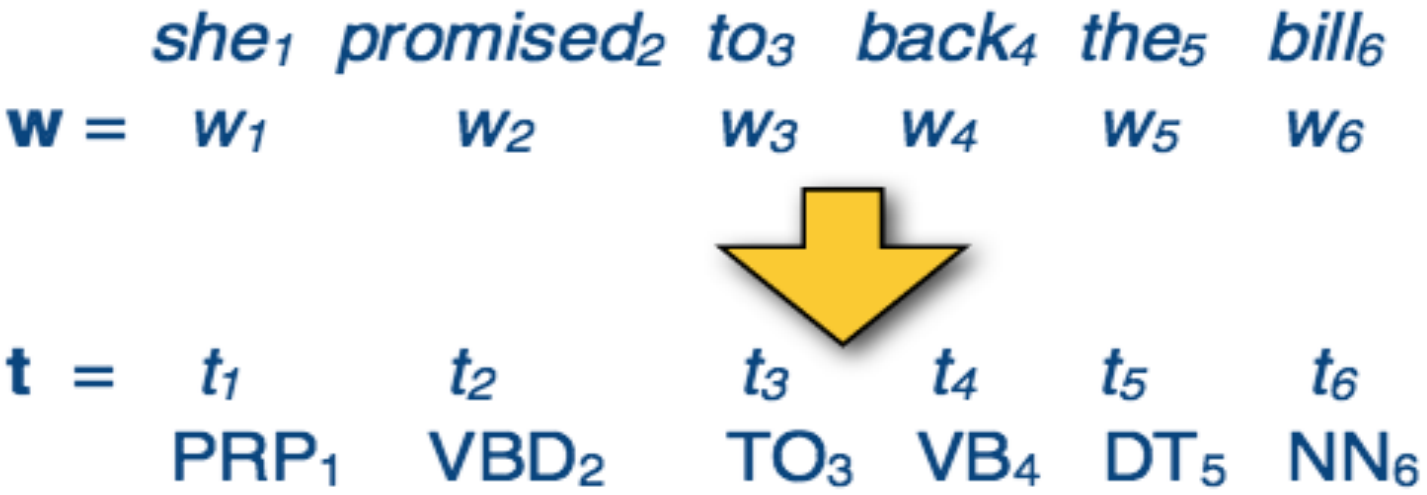
Sample HMM Generation



Sample HMM Generation



Statistical POS tagging



What is the **most likely** sequence of tags **t** for the given sequence of words **w**?

Statistical POS tagging

- What is the most likely sequence of tags t for the given sequence of words w?
  - Choosing the best tag sequence $T=t_1,t_2,\ldots,t_n$ for a given word sequence $W = w_1,w_2,\ldots,w_n$ (sentence):

$$\hat{T} = \arg\max_{T \in \tau} P(T \mid W)$$

Statistical POS tagging

- By Bayes Rule:

$$\hat{T} = \arg\max_{T \in \tau} \frac{P(W \mid T)P(T)}{P(W)}$$

- Since P(W) will be the same for each tag sequence:

$$\hat{T} = \arg\max_{T \in \tau} P(W \mid T)P(T)$$

Statistical POS tagging

- If we assume a tagged corpus and a trigram language model, then P(T) can be approximated as:

$$P(t_1)P(t_2 \mid t_1)\prod_{i=3}^{n} P(t_i \mid t_{i-2}t_{i-1})$$

- To evaluate this formula is simple, we get from simple word counting (and smoothing).

Statistical POS tagging

- To evaluate P(W|T), we will make the simplifying assumption that the word depends only on its tag:

$$\prod_{i=1}^{n} P(w_i \mid t_i)$$

Statistical POS tagging

- So, we want the tag sequence that maximizes the following quantity.

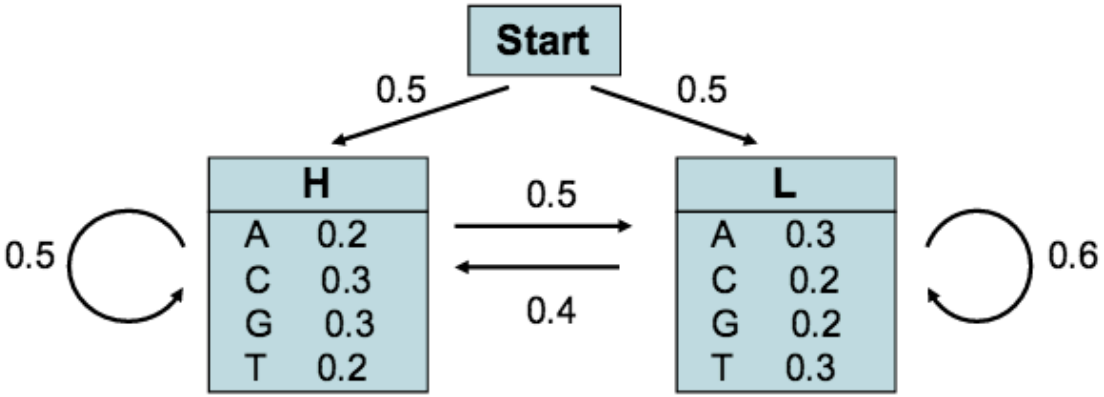
$$P(t_1)P(t_2 \mid t_1)\prod_{i=3}^{n} P(t_i \mid t_{i-2}t_{i-1})\left[\prod_{i=1}^{n} P(w_i \mid t_i)\right]$$

- The best tag sequence can be found by Viterbi algorithm.

Hidden Markov Models
(HMM's)

$$\begin{aligned}
\underset{\mathbf{t}}{\operatorname{argmax}} P(\mathbf{t}|\mathbf{w}) &= \underset{\mathbf{t}}{\operatorname{argmax}} P(\mathbf{t})P(\mathbf{w}|\mathbf{t}) \\
&:=_{def} \underset{\mathbf{t}}{\operatorname{argmax}} \underbrace{\prod_{i=1}^{n} P(t_i|t_{i-N}\ldots t_{i-1})}_{\text{Prior } P(\mathbf{t})} \underbrace{\prod_{i} P(w_i|t_i)}_{\text{Likelihood } P(\mathbf{w}|\mathbf{t})}
\end{aligned}$$

Viterbi algorithm
A toy example

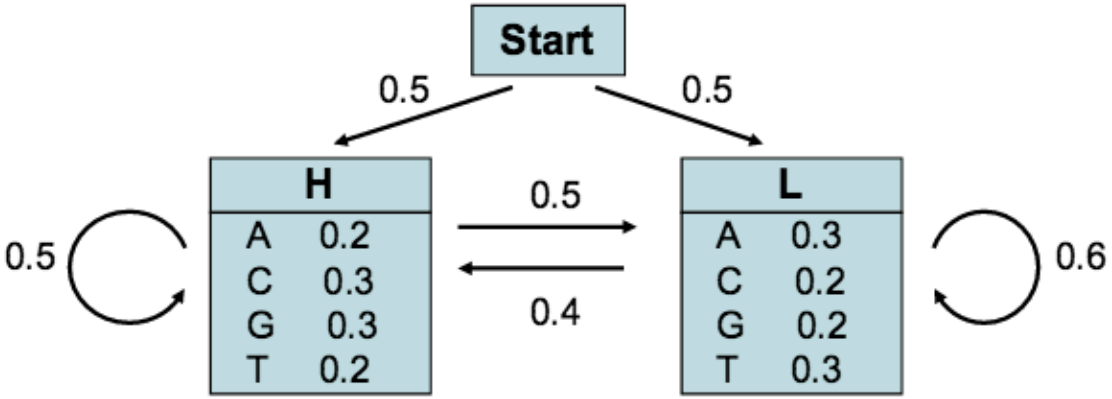


Let's consider the following simple HMM. This model is composed of 2 states, **H** (high GC content) and **L** (low GC content). We can for example consider that state H characterizes coding DNA while L characterizes non-coding DNA.

The model can then be used to predict the region of coding DNA from a given sequence.

Sources: For the theory, see Durbin *et al* (1998);
For the example, see Borodovsky & Ekisheva (2006), pp 80-81

Viterbi algorithm
A toy example



Consider the sequence S= **GGCACTGAA**

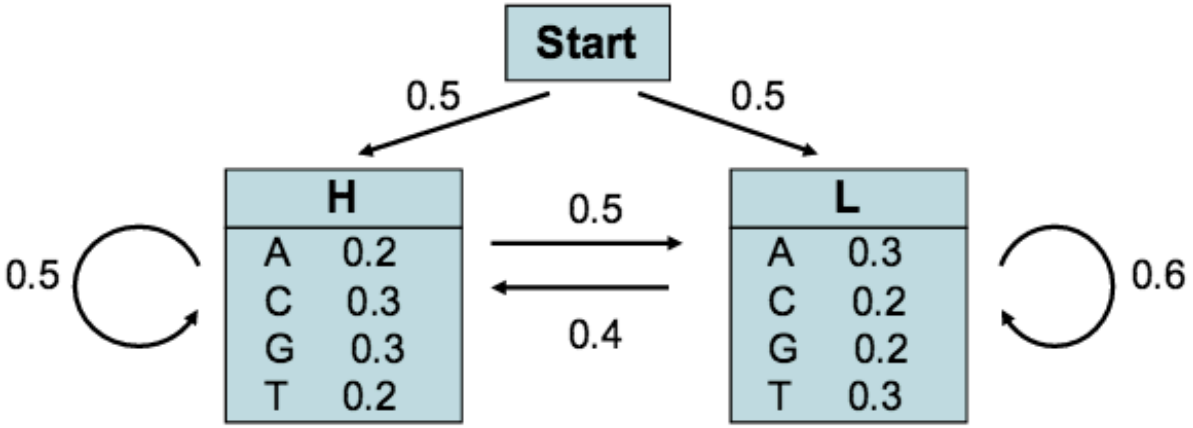
There are several paths through the hidden states (H and L) that lead to the given sequence S.

Example: P = **LLHHHHLLL**

The probability of the HMM to produce sequence S through the path P is:

$$p = p_L(0) * p_L(G) * p_{LL} * p_L(G) * p_{LH} * p_H(C) * \ldots$$
$$= 0.5 * 0.2 * 0.6 * 0.2 * 0.4 * 0.3 * \ldots$$
$$= \ldots$$

Viterbi algorithm
A toy example



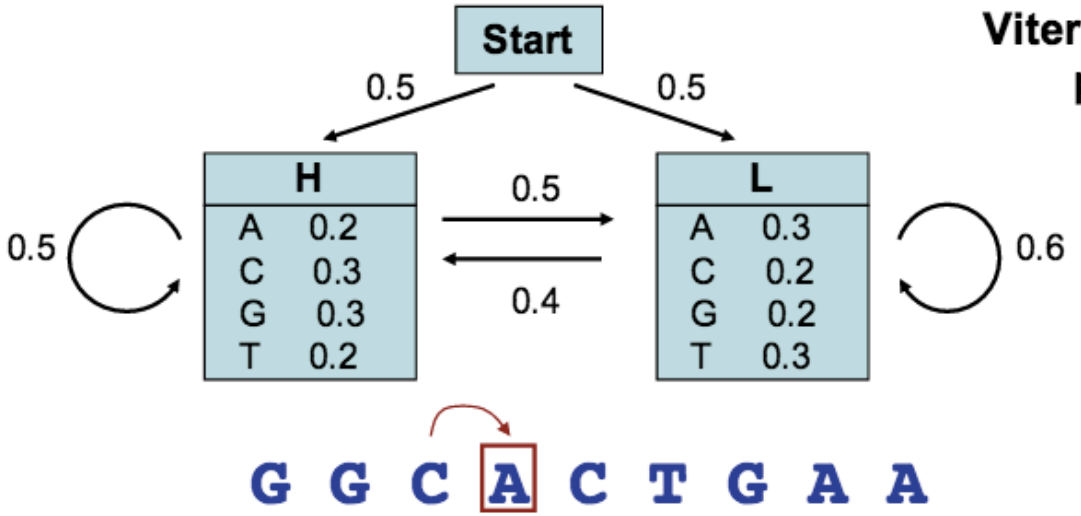
GGCACTGAA

There are several paths through the hidden states (H and L) that lead to the given sequence, but they do not have the same probability.

The **Viterbi algorithm** is a dynamical programming algorithm that allows us to compute the most probable path. Its principle is similar to the DP programs used to align 2 sequences (i.e. Needleman-Wunsch)

Source: Borodovsky & Ekisheva, 2006

Viterbi algorithm
A toy example



Viterbi algorithm: principle

The probability of the most probable path ending in state **k** with observation "i" is

$$p_l(i,x) = e_l(i)\max_k\left(p_k(j,x-1).p_{kl}\right)$$

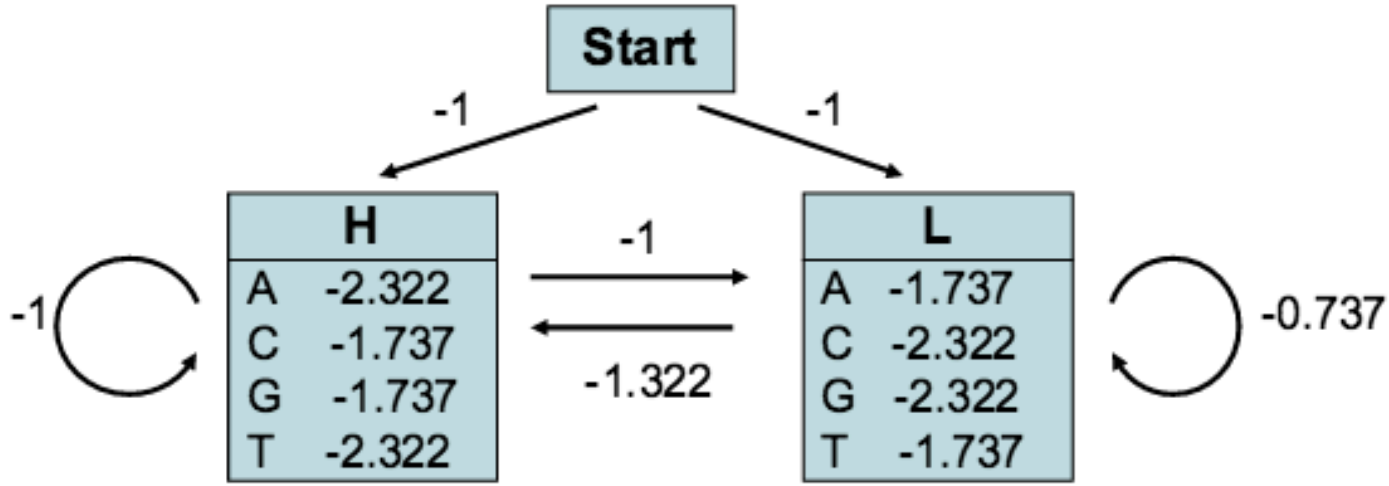
In our example, the probability of the most probable path ending in state **H** with observation "A" at the 4th position is:

$$p_H(A,4) = e_H(A)\max\left(p_L(C,3)p_{LH}, p_H(C,3)p_{HH}\right)$$

We can thus compute recursively (from the first to the last element of our sequence) the probability of the most probable path.

Viterbi algorithm

A toy example

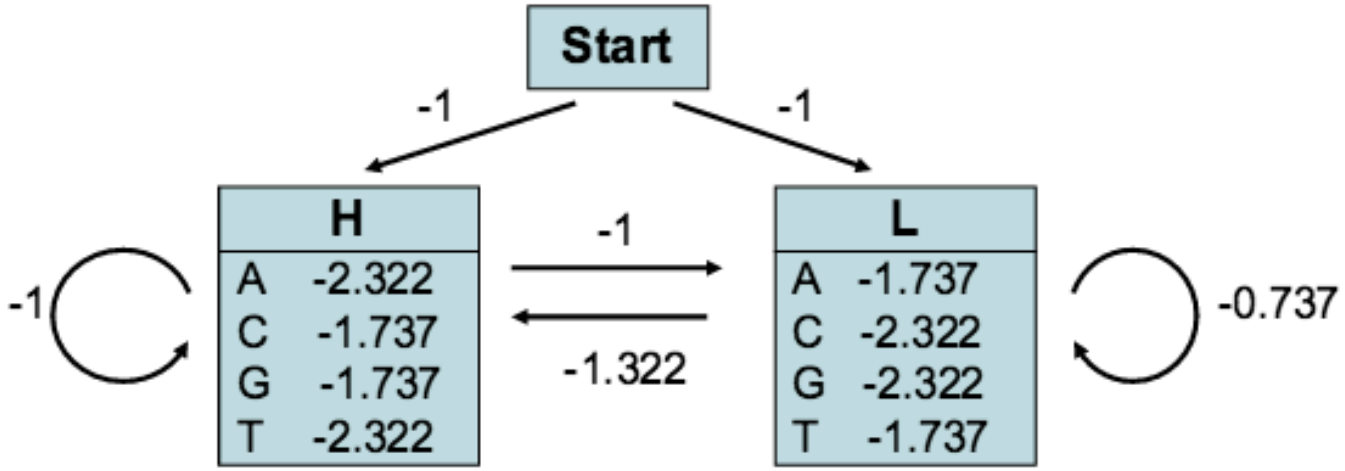


Remark: for the calculations, it is convenient to use the log of the probabilities (rather than the probabilities themselves). Indeed, this allows us to compute *sums* instead of *products*, which is more efficient and accurate.

We used here $\log_2(p)$.

Viterbi algorithm

A toy example



GGCACTGAA

Probability (in $\log_2$) that G at the first position was emitted by state **H**

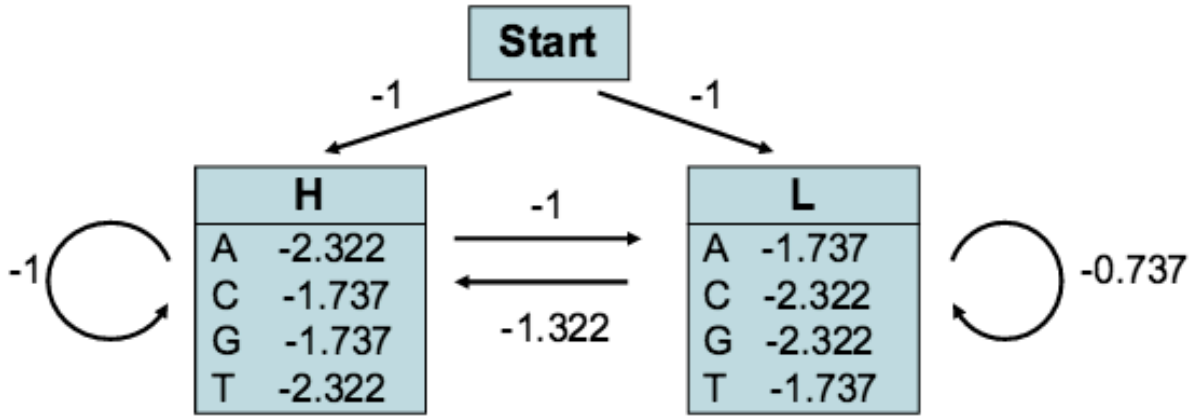
$p_H(G,1) = -1 - 1.737 = -2.737$

Probability (in $\log_2$) that G at the first position was emitted by state **L**

$p_L(G,1) = -1 - 2.322 = -3.322$

Viterbi algorithm

A toy example



GGCACTGAA

Probability (in $\log_2$) that G at the 2nd position was emitted by state **H**

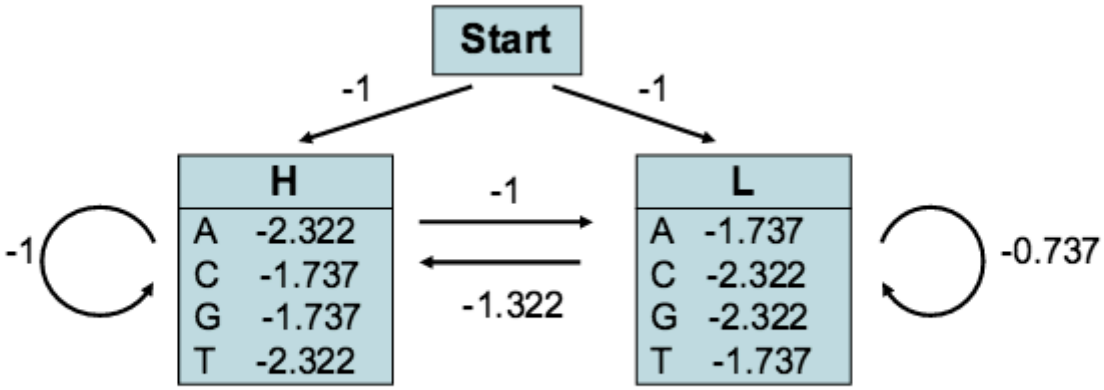
$$p_H(G,2) = -1.737 + \max(p_H(G,1)+p_{HH},\ p_L(G,1)+p_{LH})$$
$$= -1.737 + \max(-2.737 - 1,\ -3.322 - 1.322)$$
$$= -5.474 \text{ (obtained from } p_H(G,1))$$

Probability (in $\log_2$) that G at the 2nd position was emitted by state **L**

$$p_L(G,2) = -2.322 + \max(p_H(G,1)+p_{HL},\ p_L(G,1)+p_{LL})$$
$$= -2.322 + \max(-2.737 - 1.322,\ -3.322 - 0.737)$$
$$= -6.059 \text{ (obtained from } p_H(G,1))$$

Viterbi algorithm

A toy example



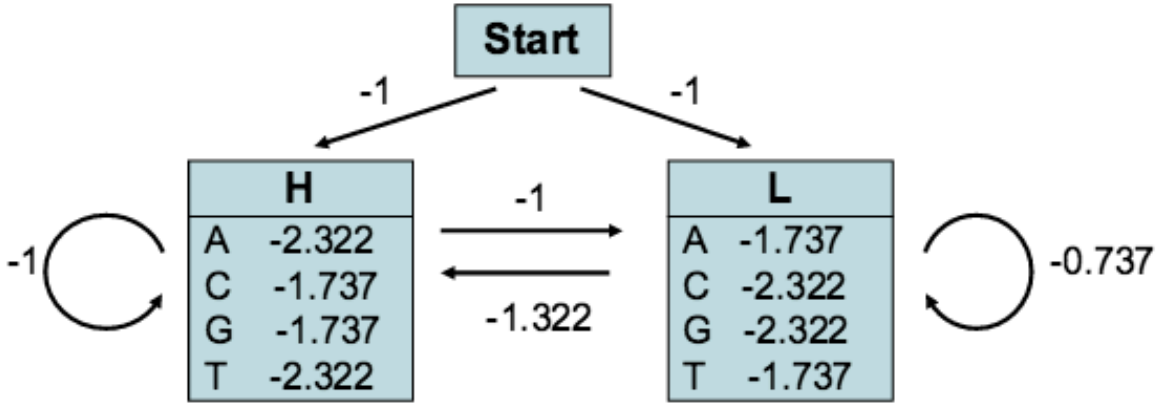
GGCACTGAA

	G	G	C	A	C	T	G	A	A
H	-2.73	-5.47	-8.21	-11.53	-14.01	...			-25.65
L	-3.32	-6.06	-8.79	-10.94	-14.01	...			-24.49

We then compute iteratively the probabilities $p_H(i,x)$ and $p_L(i,x)$ that nucleotide *i* at position *x* was emitted by state **H** or **L**, respectively. The highest probability obtained for the nucleotide at the last position is the probability of the most probable path. This path can be retrieved by back-tracking.

Viterbi algorithm

A toy example



back-tracking

(= finding the path which corresponds to the highest probability, -24.49)

GGCACTGAA

	G	G	C	A	C	T	G	A	A
H	**-2.73**	**-5.47**	**-8.21**	-11.53	-14.01	...			-25.65
L	-3.32	-6.06	-8.79	**-10.94**	**-14.01**	...			**-24.49**

The most probable path is: **HHHLLLLLLL**

Its probability is $2^{-24.49}$ = 4.25E-8
(remember that we used $\log_2(p)$)

Best Tag Sequence

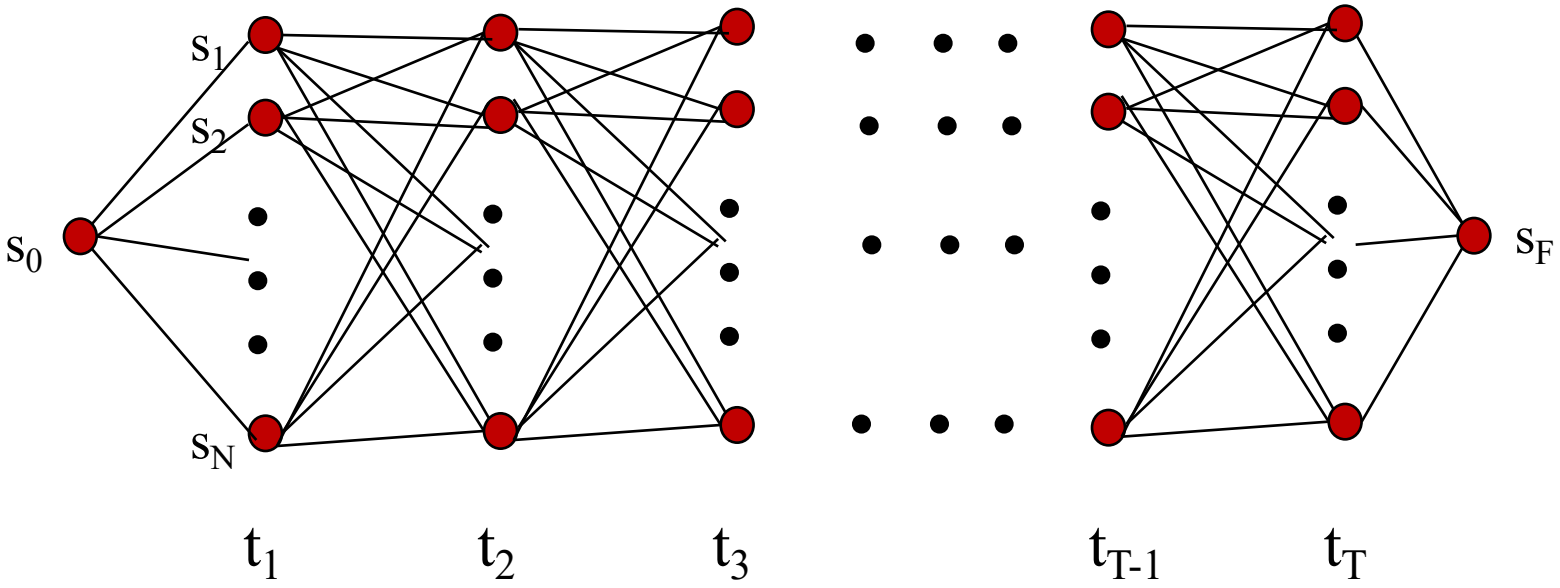
Useful problem to solve:

Given sentence, find most likely sequence of POS tags

Sue	saw	the	fly	.
noun	verb	det	noun	.

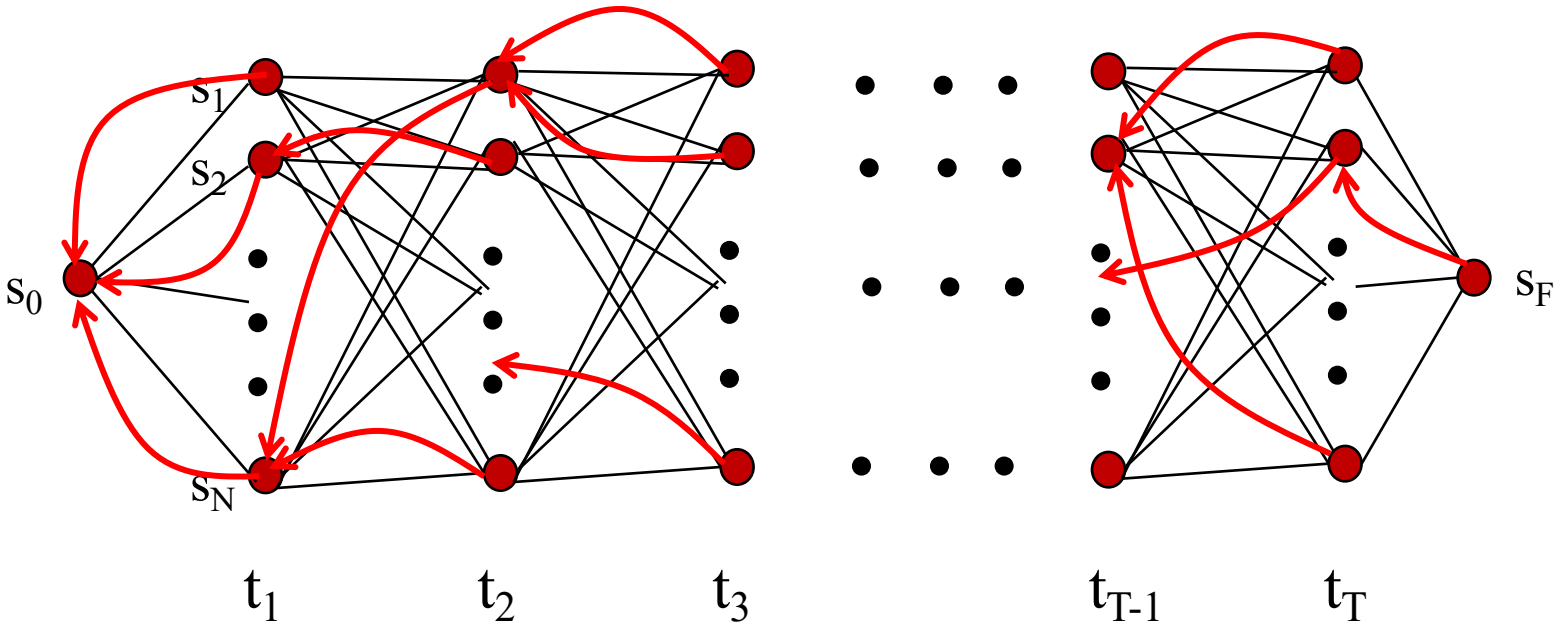
$\max_{tags}$ Pr(tags|words)

Forward Trellis

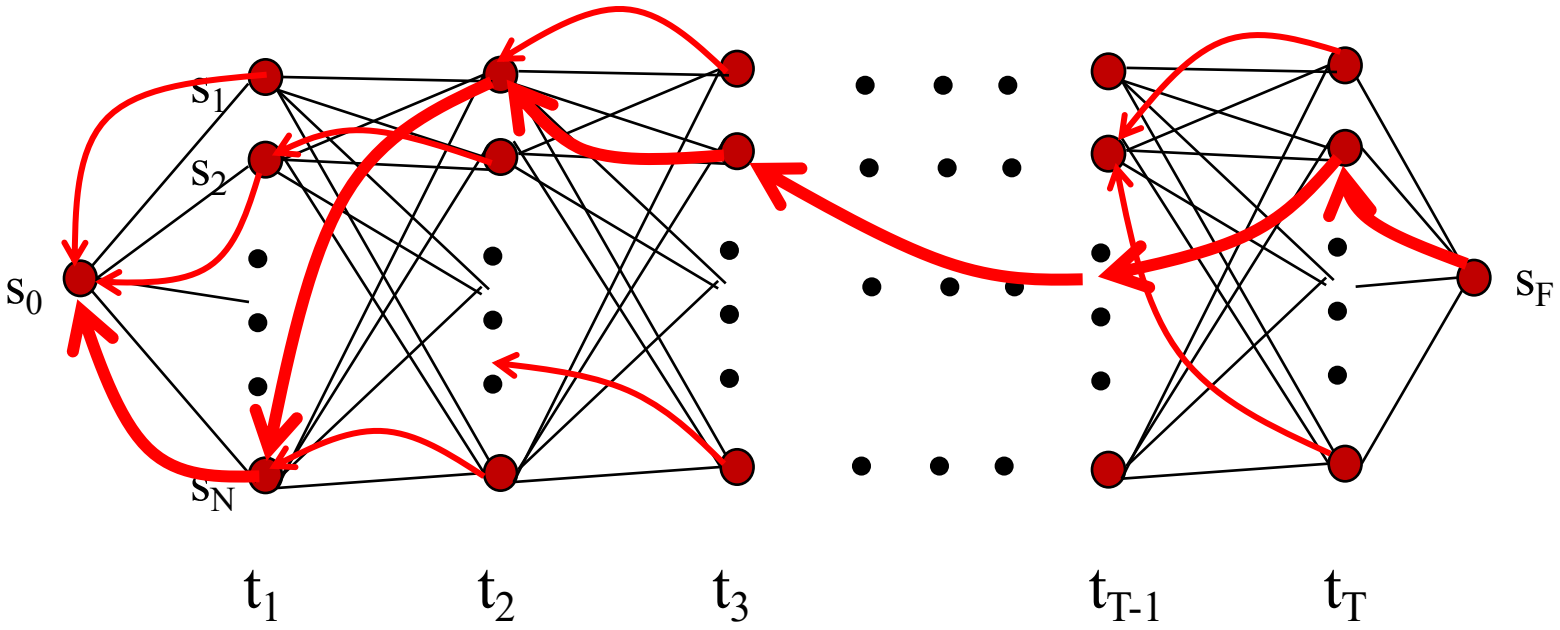


- Continue forward in time until reaching final time point.

Viterbi Backpointers



Viterbi Backtrace

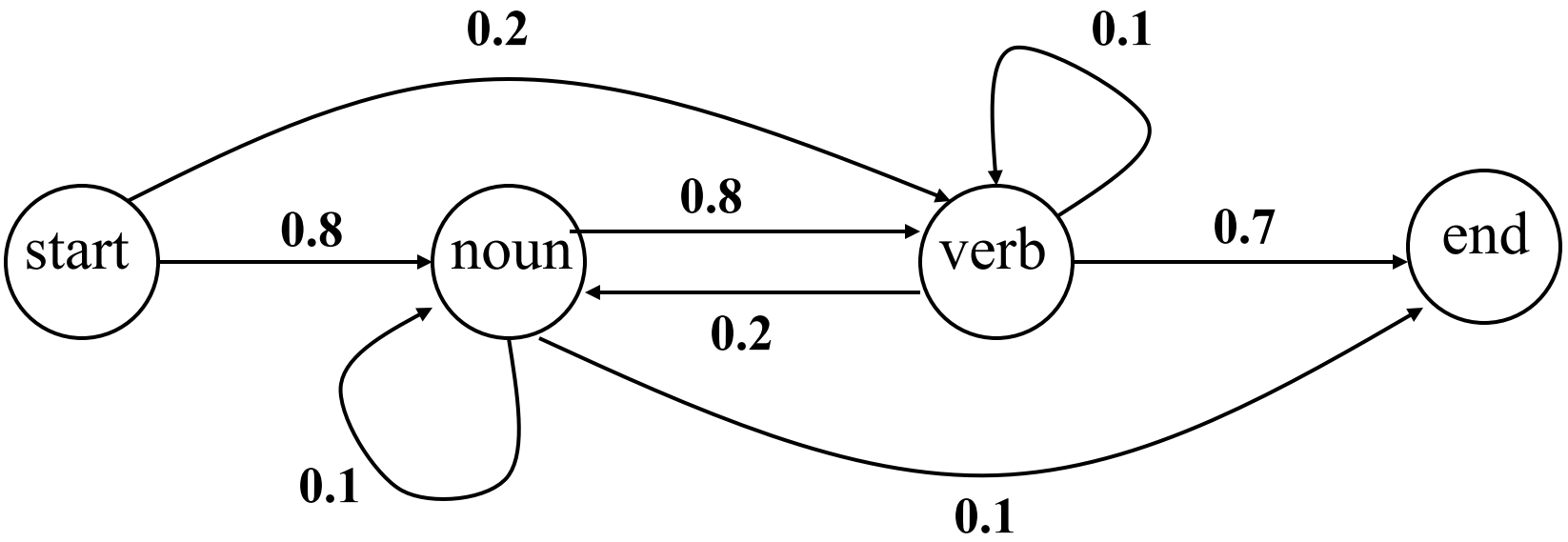


Most likely Sequence: $s_0\ s_N\ s_1\ s_2\ \ldots s_2\ s_F$

Analyzing

Fish sleep.

A Simple POS HMM

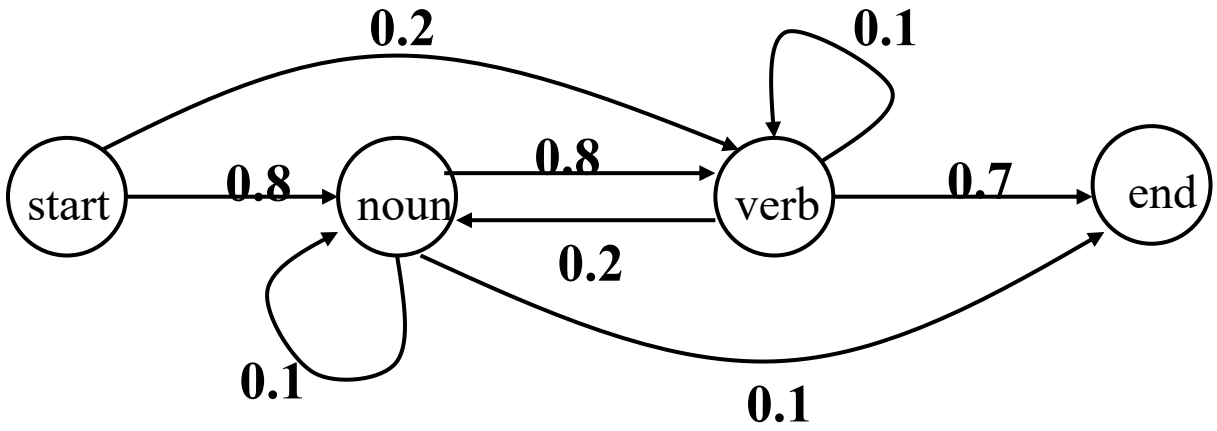


Word Emission Probabilities P (word | state)

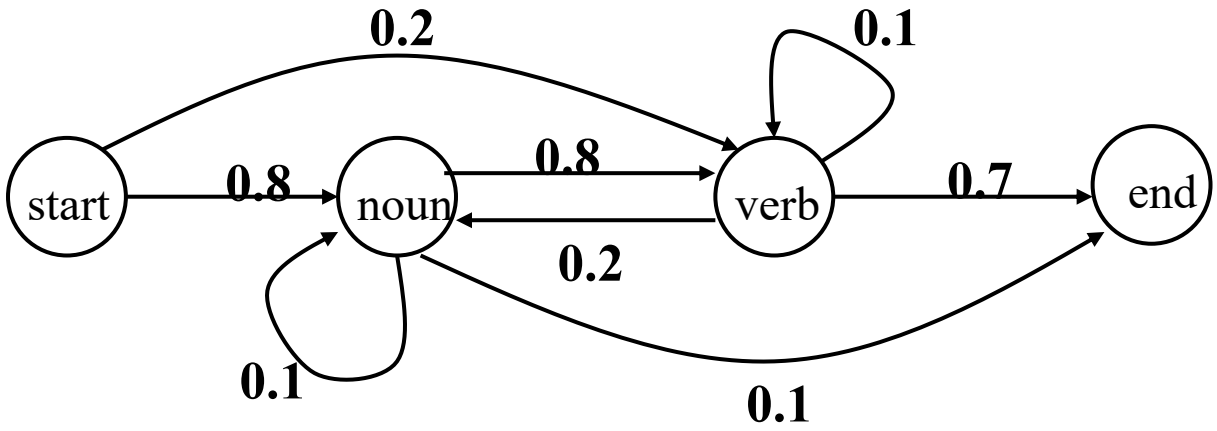
- A two-word language: "fish" and "sleep"
- Suppose in our training corpus,
  - "fish" appears 8 times as a noun and 5 times as a verb
  - "sleep" appears 2 as a noun and 5 times as a verb
- Emission probabilities:
  - Noun
    - P(fish | noun) : 0.8
    - P(sleep | noun) : 0.2
  - Verb
    - P(fish | verb) : 0.5
    - P(sleep | verb) : 0.5

Viterbi Probabilities

	0	1	2	3
start				
verb				
noun				
end				

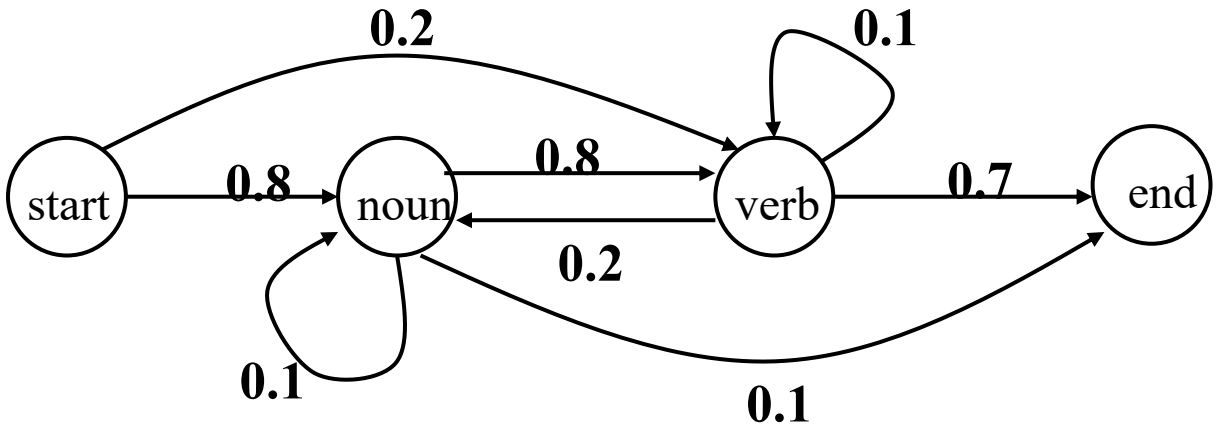


	0	1	2	3
start	1			
verb	0			
noun	0			
end	0			



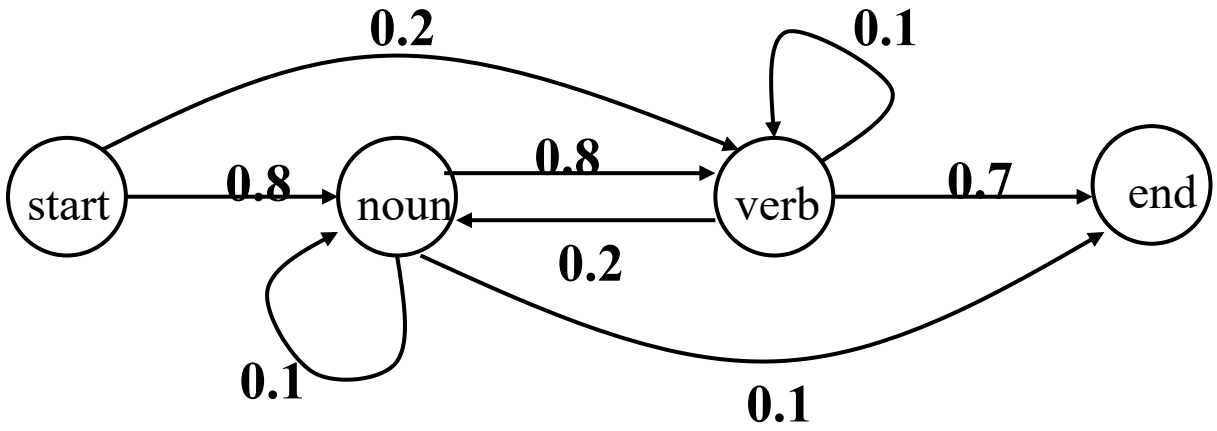
Token 1: fish

	0	1	2	3
start	1	0		
verb	0	.2 * .5		
noun	0	.8 * .8		
end	0	0		



Token 1: fish

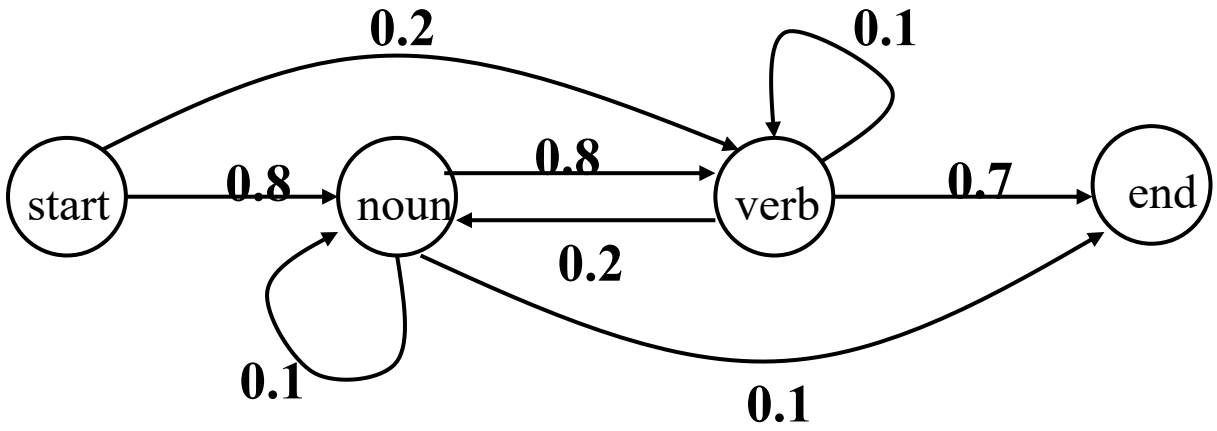
	0	1	2	3
start	1	0		
verb	0	.1		
noun	0	.64		
end	0	0		



Token 2: sleep

(if 'fish' is verb)

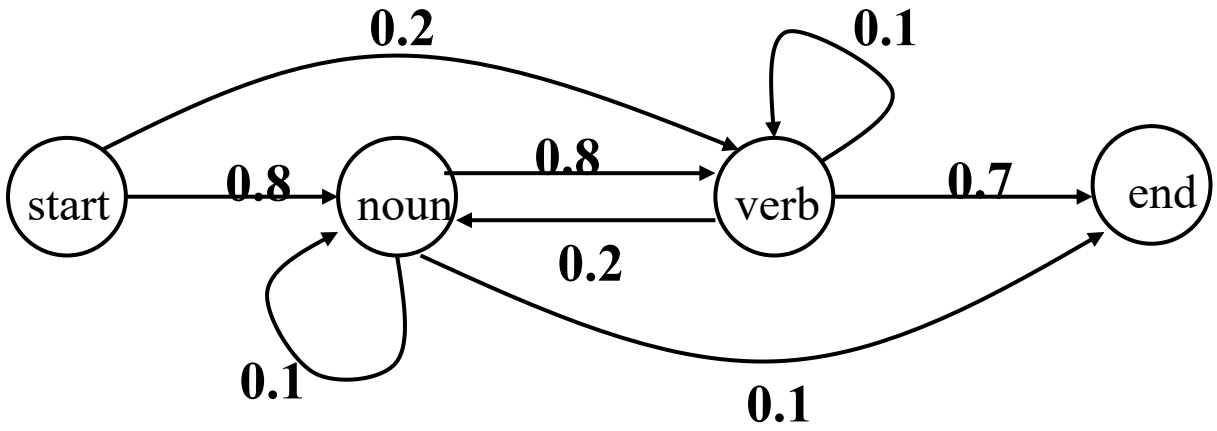
	0	1	2	3
start	1	0	0	
verb	0	.1	.1*.1*.5	
noun	0	.64	.1*.2*.2	
end	0	0	-	



Token 2: sleep

(if 'fish' is verb)

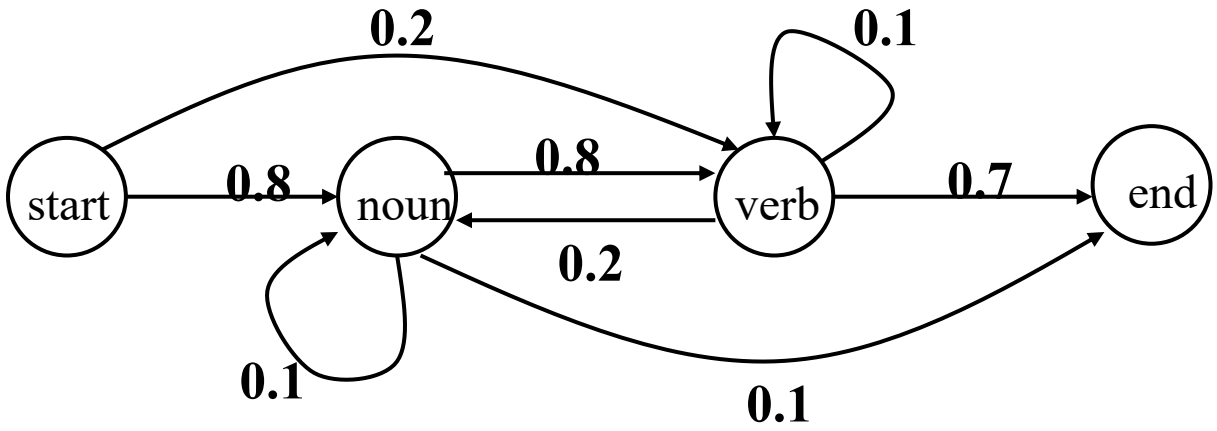
	0	1	2	3
start	1	0	0	
verb	0	.1	.005	
noun	0	.64	.004	
end	0	0	-	



Token 2: sleep

(if 'fish' is a noun)

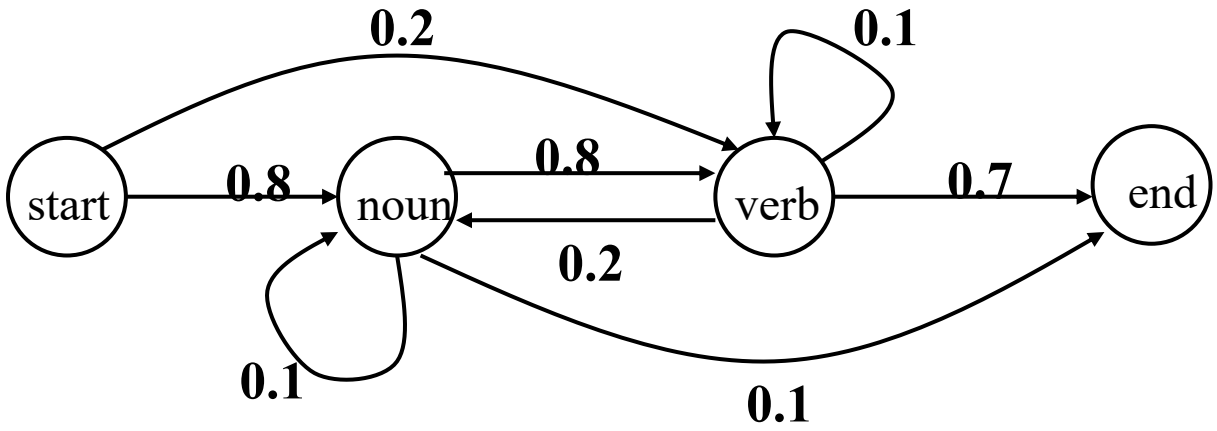
	0	1	2	3
start	1	0	0	
verb	0	.1	.005 .64*.8*.5	
noun	0	.64	.004 .64*.1*.2	
end	0	0	-	



Token 2: sleep

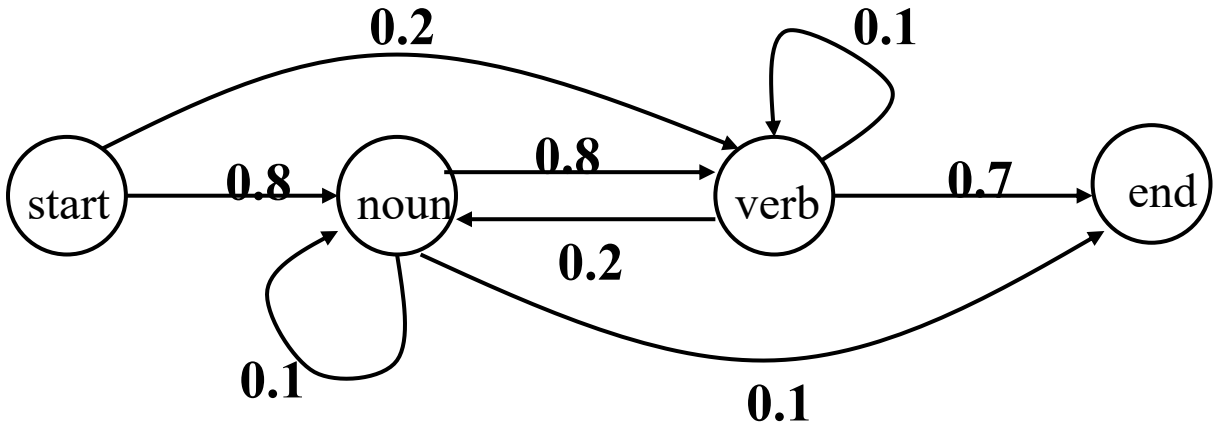
(if 'fish' is a noun)

	0	1	2	3
start	1	0	0	
verb	0	.1	.005 .256	
noun	0	.64	.004 .0128	
end	0	0	-	



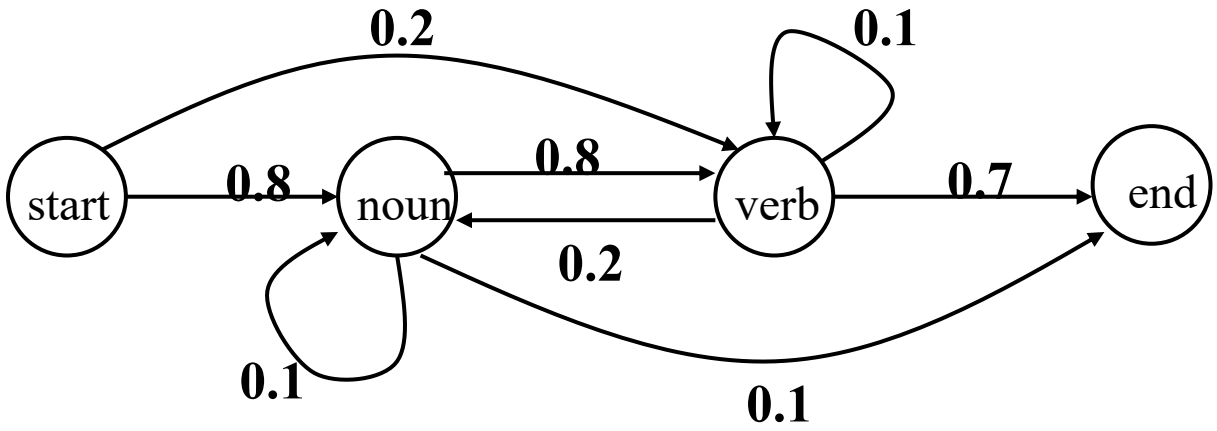
Token 2: sleep
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	
verb	0	.1	~~.005~~ .256	
noun	0	.64	~~.004~~ .0128	
end	0	0	-	



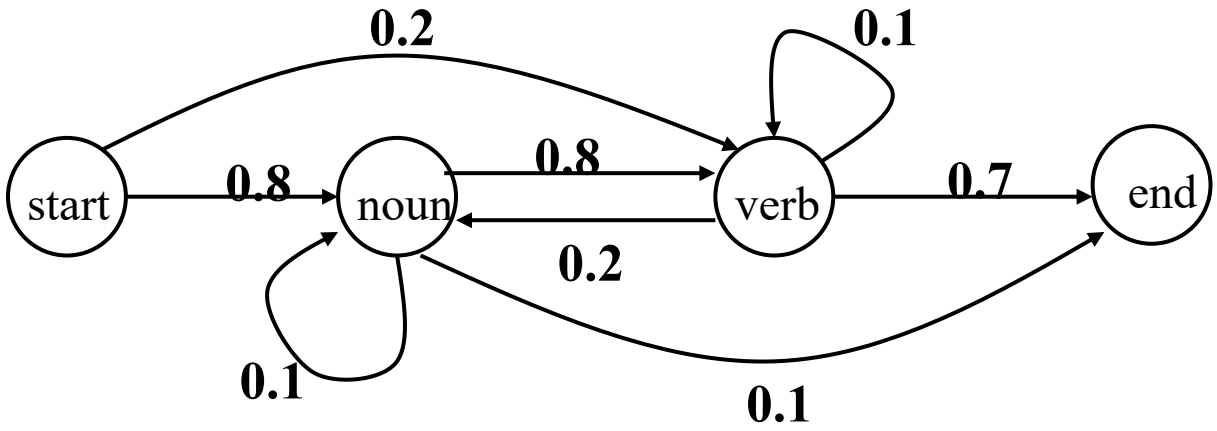
Token 2: sleep
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	
verb	0	.1	.256	
noun	0	.64	.0128	
end	0	0	-	



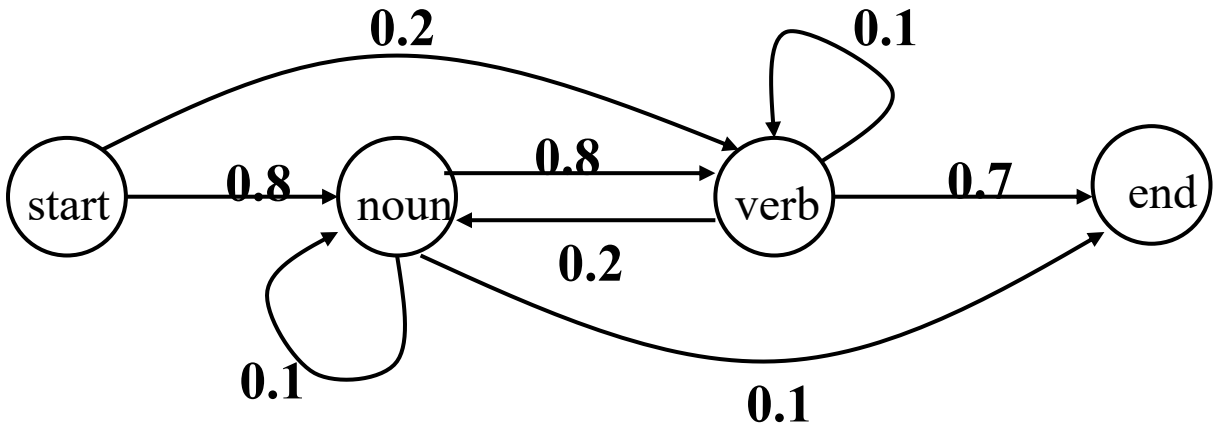
Token 3: end

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7 .0128*.1



Token 3: end
take maximum,
set back pointers

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7 ~~.0128*.1~~



Decode:
fish = noun
sleep = verb

	0	1	2	3
start	1	0	0	0
verb	0	.1	.256	-
noun	0	.64	.0128	-
end	0	0	-	.256*.7

References

- Julia Hockenmaier, PoS tagging slides.
- Jurafsky, Basic Text Processing slides