Part of speech tagging (品詞タグ付け)

Naoaki Okazaki

okazaki at ecei.tohoku.ac.jp

http://www.chokkan.org/

http://twitter.com/#!/chokkanorg

http://www.chokkan.org/lectures/2016nlp/03nlp.pdf

Today's topic

- What are part-of-speeches?
- Part-of-speech tagging as sequential labeling problem
- Hidden Markov Model (HMM)
- Structured perceptron
- Other tasks formalized as sequential labeling problem
- Implementations

Take home messages

- The *importance* and *difficulty* of annotation
- Sequential labeling problem has broad search space
  - Make use of *dynamic programing* (*Viterbi algorithm*) to find the best tag sequence
- Sequential labeling problem is easy to understand only if you could image a lattice graph
  - Mathematical formulas are (inevitably) abstract and difficult
  - Understanding structured perceptron, you are very close to Conditional Random Fields (CRFs)

Part-of-speech tags (in Penn Treebank)

Chapter 5: 5.1-5.2

Nouns (名詞)

Tag	Description	Examples
NN	Noun, singular (単数) or mass (集合)	dog, woman, snow, communism
NNS	Noun, plural (複数)	dogs, women
NP	Proper noun (固有名詞), singular	Mary, John, Sendai, Japan
NPS	Proper noun, plural	Alps, Bahamas

- Rough definitions
  - Semantically, nouns describe entities (実体) and concepts (概念)
  - Syntactically, nouns may occur with determiners (限定詞)
- Common noun
  - Count nouns (加算名詞): singular or plural
  - Mass nouns (集合名詞): singular only
    - No clear distinction from references: e.g., chair (c) and furniture (m)

Verbs (動詞)

Tag	Description	Examples
VB	Verb, base form (imperatives, infinitives, subjunctives)	eat
VBD	Verb, past tense (過去)	ate
VBG	Verb, gerund or present participle (動名詞・現在分詞)	eating
VBN	Verb, past participle (過去分詞)	eaten
VBP	Verb, non-3rd person singular present	eat
VBZ	Verb, 3rd person singular present (三人称単数現在)	eats

- Verbs refer to actions, processes, and states
- Distinction between VB and VBP
  - Imperatives (命令法): *Eat* the apple now!
  - Infinitives (不定詞): You should *eat* the apple now.
  - Subjunctives (仮定法): We suggested that he *eat* the apple.
  - VBP (三人称単数現在以外): We *eat* apples.

Adjectives (形容詞)

Tag	Description	Examples
JJ	Adjective	old, good, white, black
JJR	Adjective, comparative (比較級)	older, better
JJS	Adjective, superlative (最上級)	oldest, best

- Adjectives describe properties or quantities
- Adjectives can be:
  - attributive (or abnominal) (限定用法): modifying nouns
    - e.g., the white album
  - predicative (叙述的用法): complement (補語) of *be*
    - e.g., The album is white

Adverbs (副詞)

Tag	Description	Examples
RB	Adverb	old, good, white, black
RBR	Adverb, comparative (比較級)	older, better
RBS	Adverb, superlative (最上級)	oldest, best

- Directional (or locative): home, here, downhill
- Degree: extremely, very, somewhat
- Manner: slowly, steadily, delicately
- Temporal: yesterday, Monday

Prepositions (前置詞) and particles (不変化詞)

Tag	Description	Examples
IN	prepositions (前置詞) subordinating conjunctions (従位接続詞)	of, in, by, from after, as, because
TO	‘to’ (*regardless of prepositional or infinitive use*)	to
RB	particle (不変化詞)	up, off

- IN is ambiguous (prepositions or subordinating conjunctions)
  - Preposition: after/IN dinner/NN
  - Subordinating conjunction: after/IN the/DT party/NN ends/VBZ
  - Because Penn Treebank includes annotations of phrase structures
- Historical reason of TO
  - The Brown corpus distinguished infinitive (TO) and prepositional (IN) uses
    - To/TO access/VB to/IN the/DT repository/NN
- Particles often have extended meanings and form a phrasal verb
  - He took off/RB his hat; He took his hat off/RB.
  - She walked into/IN the room; * She walked the room into.

Other tags (1/2)

Tag	Description	Examples
CC	coordinating conjunction (等位接続詞)	and, but, or
CD	cardinal number (基数詞)	one, two, three
DT	determiner (限定詞)	a, the
EX	existential ‘there’	there
FW	foreign word	tres bien
LS	list item marker	1, 2, One
MD	modal verb (法助動詞)	can, should
PDT	predeterminer (前限定辞)	both, all, such
POS	possessive ending	‘s
PRP	personal pronoun (人称代名詞)	I, you, he, it
PRP$	possessive pronoun (所有格代名詞)	your, one’s, its

Other tags (2/2)

Tag	Description	Examples	Note
SYM	symbol	+, %, &	
UH	interjection (間投詞，感嘆詞）	ah, oops	
WDT	wh-determiner	which, that	preceding nouns (*what* kind of …)
WP	wh-pronoun	what, who	(Tell me *what* you think)
WP$	possessive wh-	whose	
WRB	wh-adverb	how, where	*How* long did you stay?
$	dollar sign	$	
#	pound sign	#	
“	left quote	` or “	
”	right quote	’ or ”	
(	left parenthesis (開き括弧)	[({ <	
)	right parenthesis (閉じ括弧)	]) } >	
,	comma	,	
.	sentence final punctuation	. ! ?	
:	mid-sentence punctuation	: ; … -	

Exercise 1: Annotate underlined words

- The near side of the moon.
- They had approached quite near.
- We were near the station.
- I don’t trust him at all.
- All right!
- The couple loves each other.
- He wore a striking hat.
- The striking teachers protested.
- The reading for this class is difficult.
- He was invited by some friends of hers.
- He was very surprised by her remarks.

Answer 1: Annotate underlined words

- The near side of the moon.
- They had approached quite near.
- We were near the station.
- I don't trust him at all.
- All right!
- The couple loves each other.
- He wore a striking hat.
- The striking teachers protested.
- The reading for this class is difficult.
- He was invited by some friends of hers.
- He was *very* surprised by her remarks.

OK

Penn Treebank POS guideline (Santorini, 1990)

- Example: JJ (alternatively, VBG)?
  - Gradable: can be preceded by very, or allows comparative
    - Her talk was very *interesting*.
    - Her talk was more *interesting* than theirs.
  - Prefix un- with the opposite meaning
    - An *interesting* conversation
    - An un*interesting* conversation
  - Construction with *be*, and be could *be* replaced with *become*, etc.
    - The conversation became depressing.
    - That place sound depressing.
  - Precedes noun, the corresponding verb is intransitive (自動詞), or does not have the same meaning
    - A *striking* hat (* the hat will strike)
    - The *striking* teachers (the teachers will strike)

Annotation work

- Annotation work requires an *annotation guideline* to ensure consistent annotations
  - Specification (e.g., definitions of annotations)
  - Tests for confusing examples

- We seldom know the ‘best’ way to annotate in advance
  - Exceptions
  - Conflicts between application and linguistic points of view
  - We often revise an annotation guideline during annotation work

- We often ask multiple humans to annotate the same text
  - Annotation agreement (kappa) to assess annotation consistency and quality

Part-of-Speech Tagging as Sequential Labeling Problem

Chapter 5: 5.3

Sequential Labeling Problem (系列ラベリング問題)

- A given sentence, *"Time flies like an arrow"*
- Represent the input sentence with a token vector $\boldsymbol{x}$

t	1	2	3	4	5
$\boldsymbol{x}$	$Time$	$flies$	$like$	an	$arrow$
	x_1	x_2	x_3	x_4	x_5

(T = 5)

(Bold italic)

(*NOTE: This does not present a feature vector*)

- Predict part-of-speech (a vector $\boldsymbol{y}$) tags for the tokens $\boldsymbol{x}$

t	1	2	3	4	5
$\boldsymbol{x}$	$Time$	$flies$	$like$	an	$arrow$
	x_1	x_2	x_3	x_4	x_5
$\boldsymbol{y}$	NN	VBZ	IN	DT	NN
	y_1	y_2	y_3	y_4	y_5

Predict

Point-wise vs sequential-wise labeling

- Point-wise labeling: predict a part-of-speech tag for each token independently

$$\left.\begin{array}{cccccc} t & 1 & 2 & 3 & 4 & 5 \\ \boldsymbol{x} & Time & flies & like & an & arrow \\ & x_1 & x_2 & x_3 & x_4 & x_5 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \boldsymbol{y} & NN & VBZ & IN & DT & NN \\ & y_1 & y_2 & y_3 & y_4 & y_5 \end{array}\right\}$$

Predict y_1 from x_1, y_2 from x_2, …, y_5 from x_5, independently

  - We can apply linear classifiers multiple times for this approach

- However, this approach cannot incorporate dependence of predicted labels, e.g., "VB* usually follow DT or NN*"
- Besides, decisions at every token may be inconsistent

- *Sequential-wise labeling*: predict a *sequence* of tags $\boldsymbol{y}$ from an input *sequence* of tokens $\boldsymbol{x}$ *at a time*!

Ambiguity and disambiguation

- Ambiguity of POS tags
  - Several assignments of POS tags are plausible for the example

t	1	2	3	4	5	
$\boldsymbol{x}$	$Time$	$flies$	$like$	an	$arrow$	
$\boldsymbol{y}_1$	NN	VBZ	IN	DT	NN	（光陰矢のごとし）
$\boldsymbol{y}_2$	VB	NNS	IN	DT	NN	（ハエの速度を矢のように測定せよ）
$\boldsymbol{y}_3$	NN	NNS	VBP	DT	NN	（時蠅は矢を好む）

- Disambiguation (resolving ambiguity) of POS tags
  - Probabilistic approach: to find the best POS tag sequence of all possible sequences by using a conditional probability (scoring)

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, P(\boldsymbol{y}|\boldsymbol{x})$$

$\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \ldots$

$\hat{\boldsymbol{y}}$ means "our estimation for $\boldsymbol{y}$"

argmax: find $\boldsymbol{y}$ that maximizes $P(\boldsymbol{y}|\boldsymbol{x})$

Three things need to be considered

- *Modeling*: how to build (assume) $P(\boldsymbol{y}|\boldsymbol{x})$
  - Hidden Markov Model (HMM), Structured Perceptron, Conditional Random Fields (CRFs), etc

- *Training*: how to determine unknown parameters in the model so that they fit to a training data
  - Maximum Likelihood (ML), Maximum a Posteriori (MAP), etc
  - Gradient-based method, Stochastic Gradient Descent (SGD), etc

- *Inference*: how to compute $\mathrm{argmax}\, P(\boldsymbol{y}|\boldsymbol{x})$ efficiently
  - Viterbi algorithm

Part-of-speech tagging using Hidden Markov Model (HMM)

Chapter 5: 5.5

Modeling

- Recap of the formalization: we want to model $P(\boldsymbol{y}|\boldsymbol{x})$
  - $\boldsymbol{x}$: the sequence of tokens (words)
  - $\boldsymbol{y}$: the sequence of POS tags
- Bayes' theorem:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})}{P(\boldsymbol{x})}$$

- Bayesian inference: decompose $P(\boldsymbol{y}|\boldsymbol{x})$ into two factors, $P(\boldsymbol{x}|\boldsymbol{y})$ and $P(\boldsymbol{y})$, which might be easier to model

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \underset{\substack{\uparrow \\ \text{Bayes' theorem}}}{=} \operatorname*{argmax}_{\boldsymbol{y}} \frac{P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})}{P(\boldsymbol{x})} \underset{\substack{\uparrow \\ P(\boldsymbol{x}) \text{ is the same for all } \boldsymbol{y}}}{=} \operatorname*{argmax}_{\boldsymbol{y}} P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})$$

Modeling (cont'd)

- Two Markov assumptions to simplify $P(\boldsymbol{x}|\boldsymbol{y})$ and $P(\boldsymbol{y})$
  - A word appears depending only on its POS tag
    - Independently of other words around the word
    - Generated by emission probability distribution

$$P(\boldsymbol{x}|\boldsymbol{y}) \approx \prod_{t=1}^{T} P(x_t|y_t)$$

  - A POS tag is dependent only on the previous one
    - Rather than the entire tag sequence
    - Generated by transition probability distribution

$$P(\boldsymbol{y}) \approx \prod_{t=1}^{T} P(y_t|y_{t-1})$$

- Then, the most probable tag sequence $\hat{\boldsymbol{y}}$ is computed by,

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, P(\boldsymbol{y}|\boldsymbol{x}) = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y}) \approx \underset{\boldsymbol{y}}{\operatorname{argmax}} \prod_{t=1}^{T} P(x_t|y_t)P(y_t|y_{t-1})$$

- In other words, we find the most probable tag sequence that maximizes the function $\phi(\boldsymbol{x}, \boldsymbol{y})$ (instead of $P(\boldsymbol{y}|\boldsymbol{x})$),

$$\phi(\boldsymbol{x}, \boldsymbol{y}) \equiv \prod_{t=1}^{T} P(x_t|y_t)P(y_t|y_{t-1})$$

Training

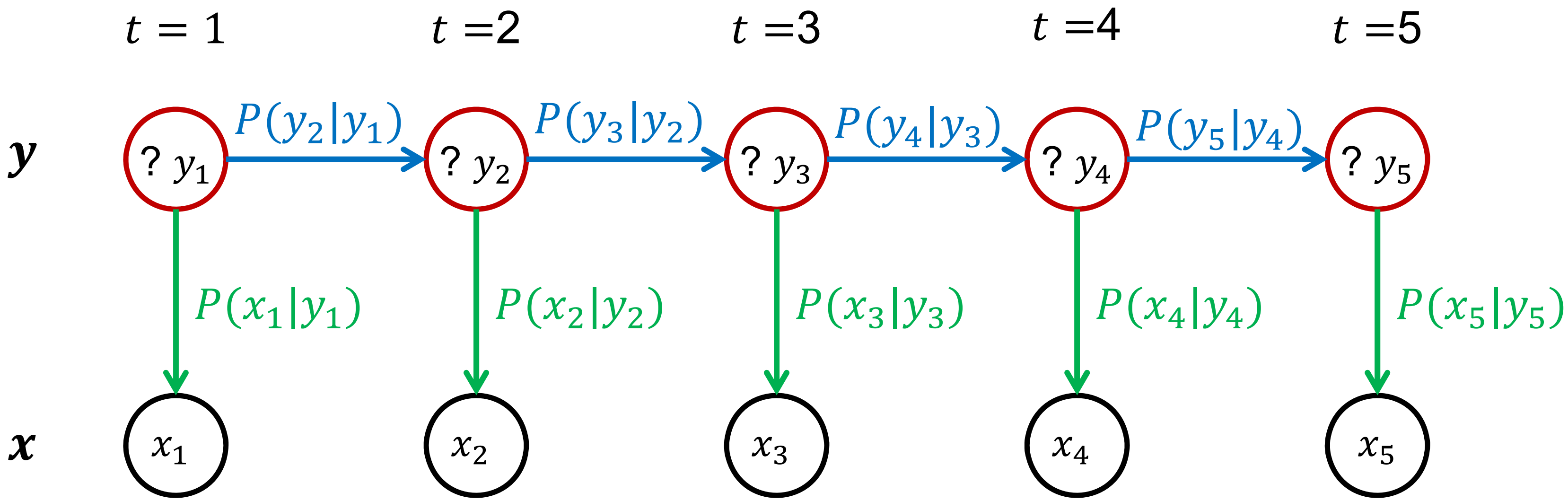
- Maximum likelihood estimation (最尤推定)

$$P(x_t|y_t) = \frac{C(x_t, y_t)}{C(y_t)} = \frac{(\text{the number of times where } x_t \text{ is annotated as } y_t)}{(\text{the number of occurrences of tag } y_t)}$$

$$P(y_t|y_{t-1}) = \frac{C(y_t, y_{t-1})}{C(y_{t-1})} = \frac{(\text{the number of occurrences of tag } y_t \text{ followd by } y_{t-1})}{(\text{the number of occurrences of tag } y_{t-1})}$$

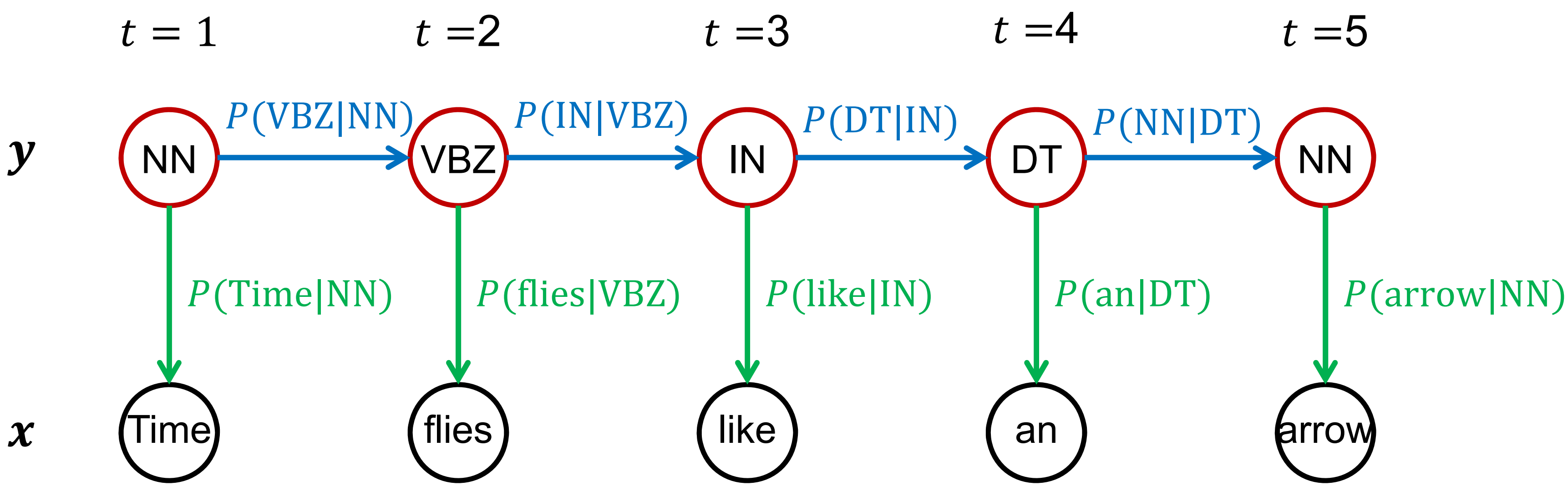
- As simple as counting frequency of co-occurrences in a training set!
- (See the appendix)

Graphical representation of $\phi(\boldsymbol{x}, \boldsymbol{y})$



- Hidden Markov Model (HMM) (隠れマルコフモデル), 1st order linear-chain (一次線形連鎖)
  - We can view this model generates a sequence of (observable) words $\boldsymbol{x}$ from hidden (unobservable) states $\boldsymbol{y}$

Computing $\phi(\boldsymbol{x}, \boldsymbol{y})$



- We can compute $\phi(\boldsymbol{x}, \boldsymbol{y})$ if we decide an assignment of $\boldsymbol{y}$ for a given input $\boldsymbol{x}$: $\prod_{t=1}^{T} P(x_t|y_t)P(y_t|y_{t-1})$

- The POS tagging task estimates the most probable hidden states $\hat{\boldsymbol{y}}$ that yields the highest $\phi(\boldsymbol{x}, \boldsymbol{y})$

A toy example: “Brown promises free”

- A very restricted language
  - Three tokens only: “Brown”, “promises”, and “free”
  - Three part-of-speeches only: noun, verb, adjective (adj)

- Suppose that probabilistic distributions estimated…

Emission $P(x_t|y_t)$

	Brown	promises	free
Noun	0.3	0.3	0.4
Verb	0.2	0.4	0.4
Adj	0.5	0.1	0.4

Transition $P(y_t|y_{t-1})$

	Noun	Verb	Adj
Noun	0.4	0.5	0.1
Verb	0.7	0.1	0.2
Adj	0.5	0.2	0.3

Exercise 2: computing $\phi(\boldsymbol{x}, \boldsymbol{y})$

- Compute $\phi(\boldsymbol{x}, \boldsymbol{y})$ for:
  - 1) Brown/adj promises/noun free/verb
  - 2) Brown/noun promises/verb free/noun

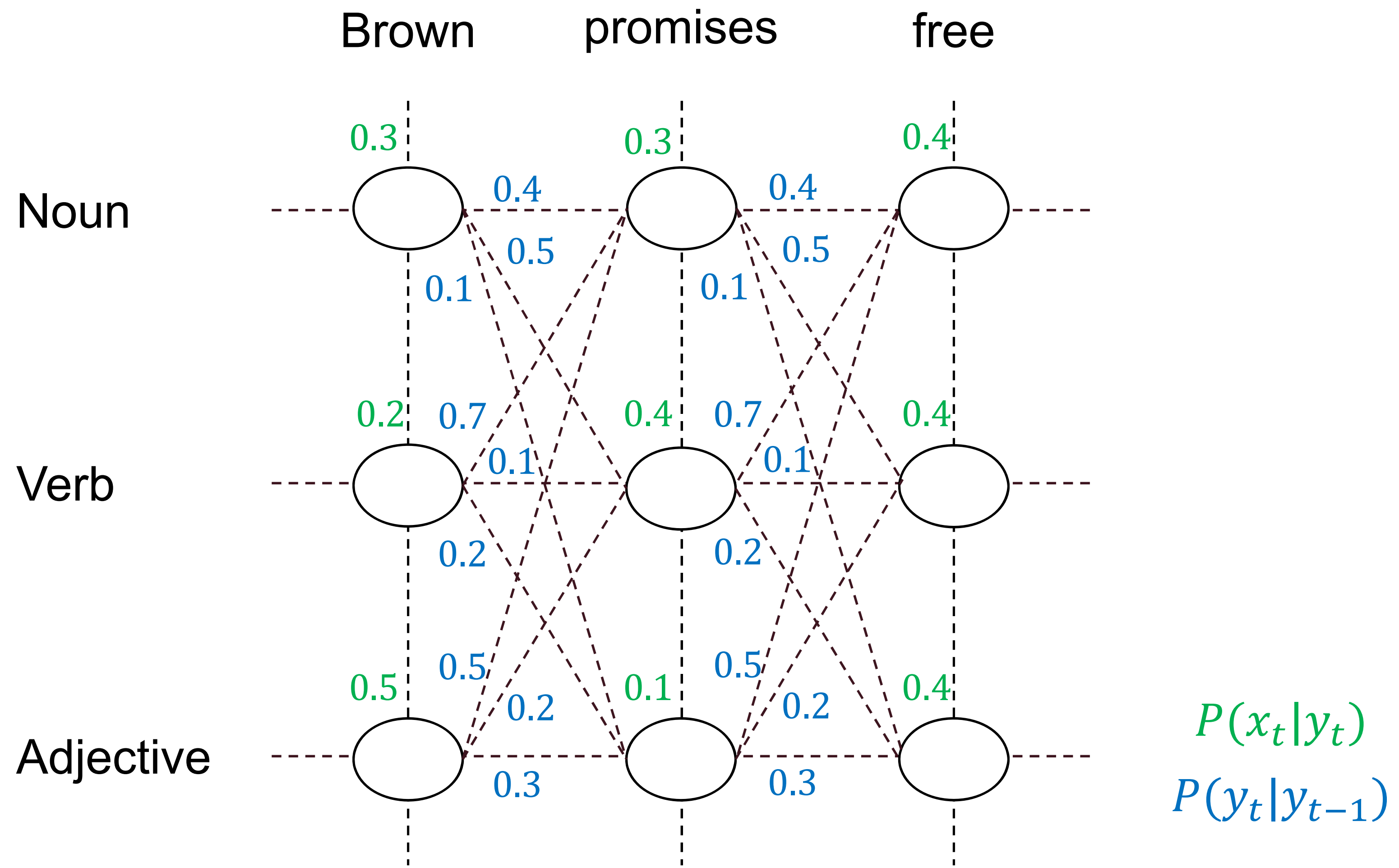
Answer 2: computing $\phi(\boldsymbol{x}, \boldsymbol{y})$

Inference: $\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\mathrm{argmax}} \prod_{t=1}^{T} P(x_t|y_t)P(y_t|y_{t-1})$

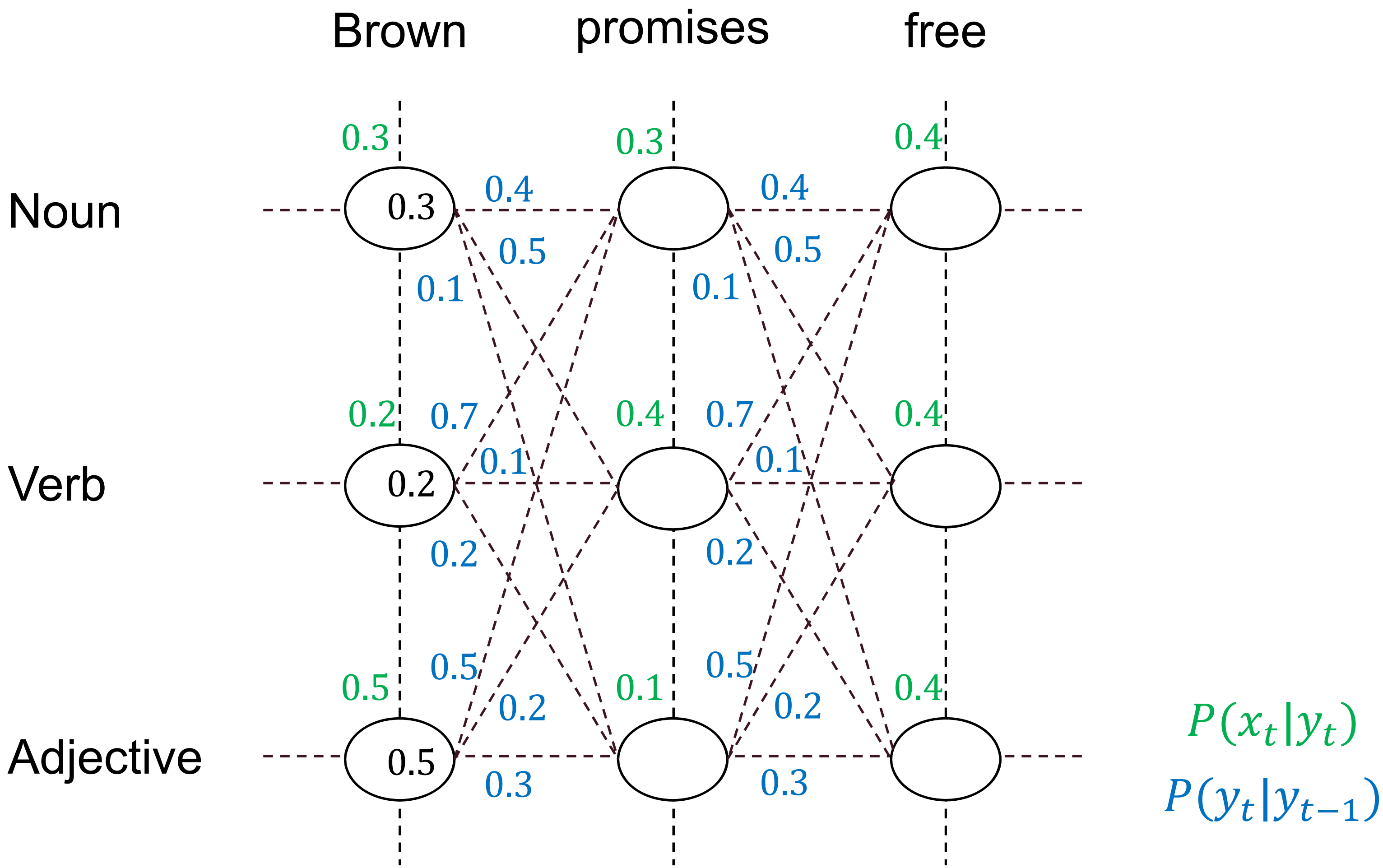
- We cannot enumerate all possible $\boldsymbol{y}$ for an input $\boldsymbol{x}$
  - The number of candidate sequences is $|Y|^T$, where:
    - $|Y|$: the number of POS tags ($|Y| = 36$ for Penn Treebank)
    - T: the number of tokens in an input sentence
  - The number of candidates is too huge, $36^6 = 2176782336$, even for the short example sentence!
- Viterbi algorithm
  - An efficient algorithm for finding $\hat{\boldsymbol{y}}$
  - Computational cost: $O(|Y|^2 T)$
  - Dynamic programing (動的計画法)
    - Dijkstra's algorithm
    - Max-product algorithm

Andrew Viterbi

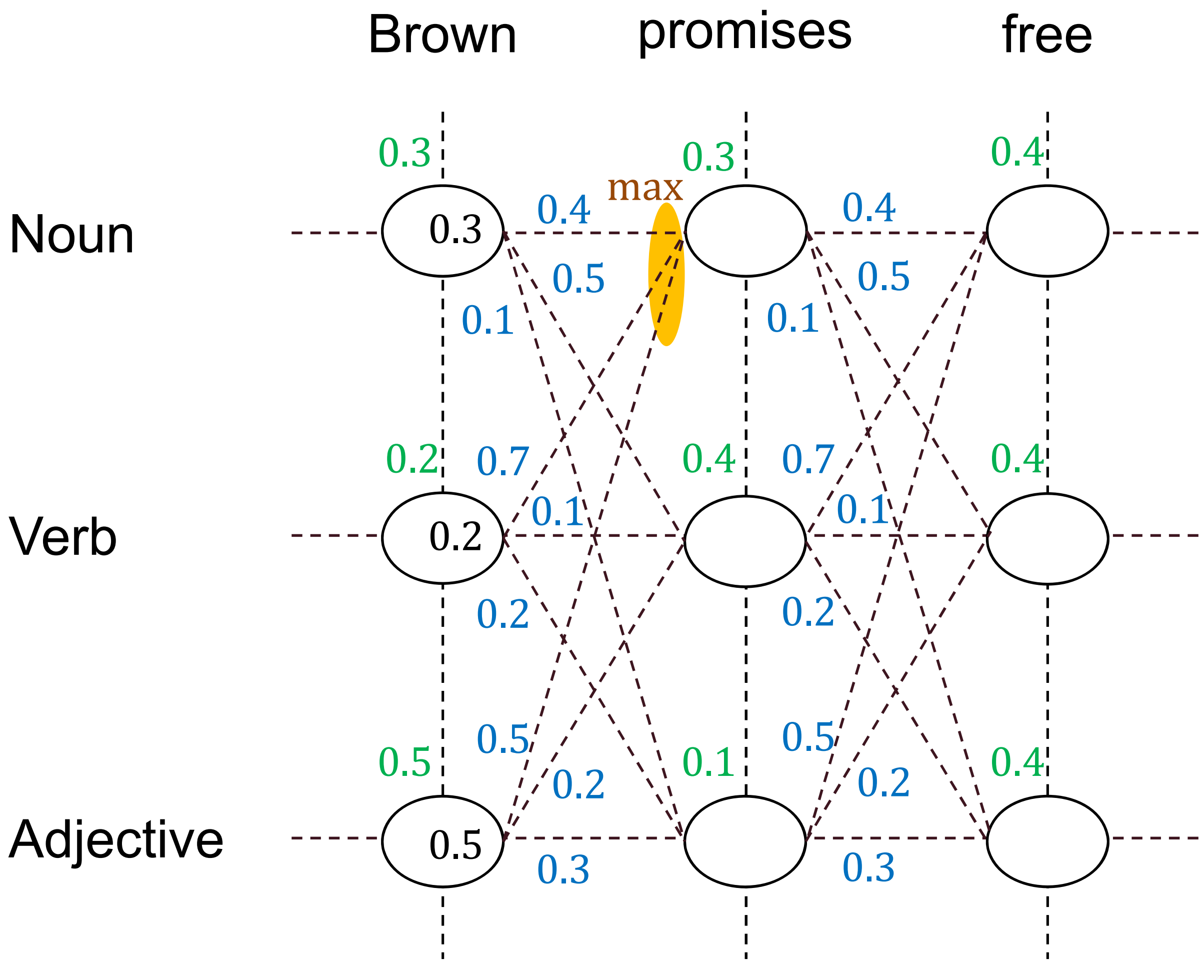
Viterbi (0/11) – Lattice representation



Viterbi (1/11) – Initialization at $t = 1$



Viterbi (2/11) – Max route to noun at $t = 2$

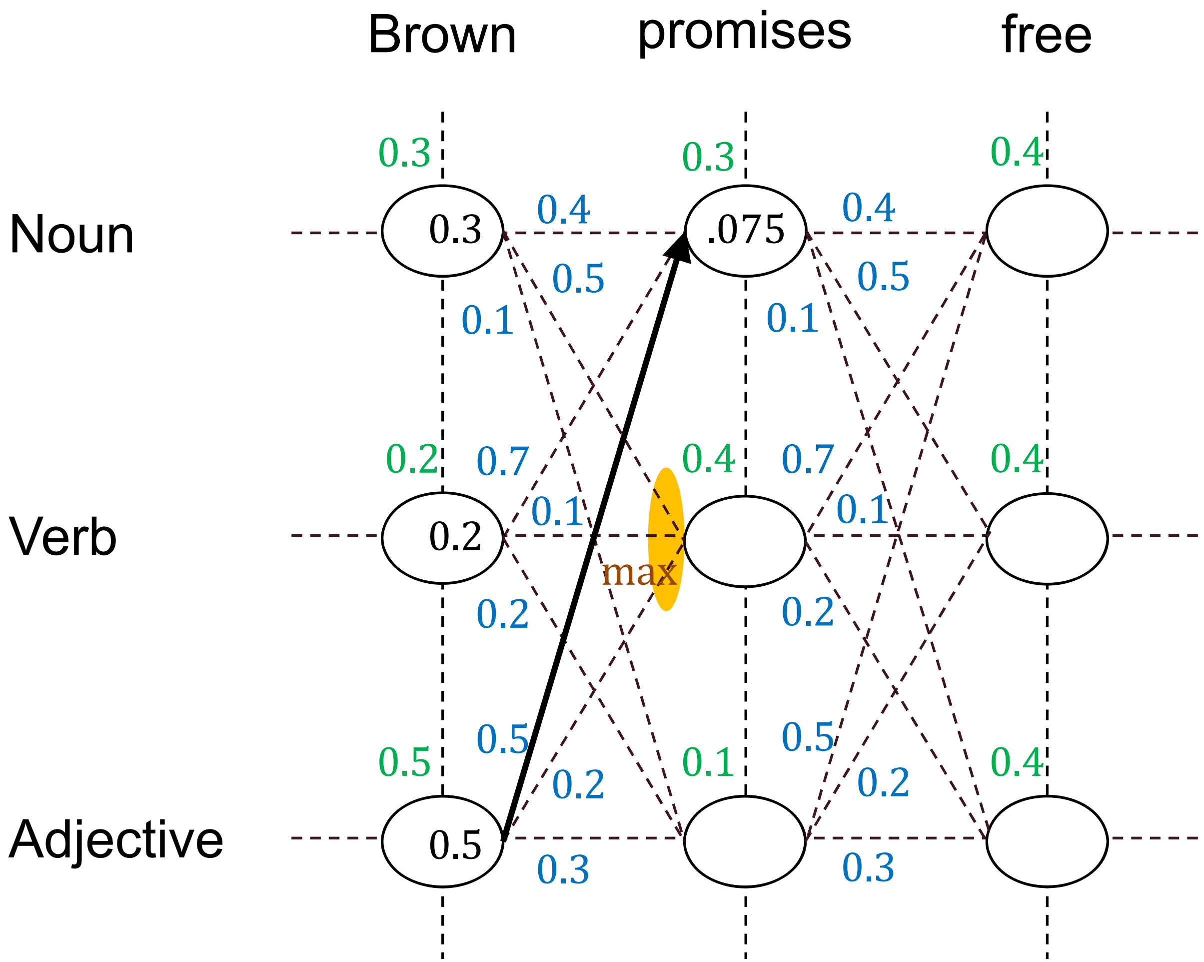


Noun - Noun
$0.3 \times 0.4 \times 0.3 = 0.036$

Verb - Noun
$0.2 \times 0.7 \times 0.3 = 0.042$

Adj - Noun
$0.5 \times 0.5 \times 0.3 = 0.075$

Viterbi (3/11) – Max route to verb at $t = 2$

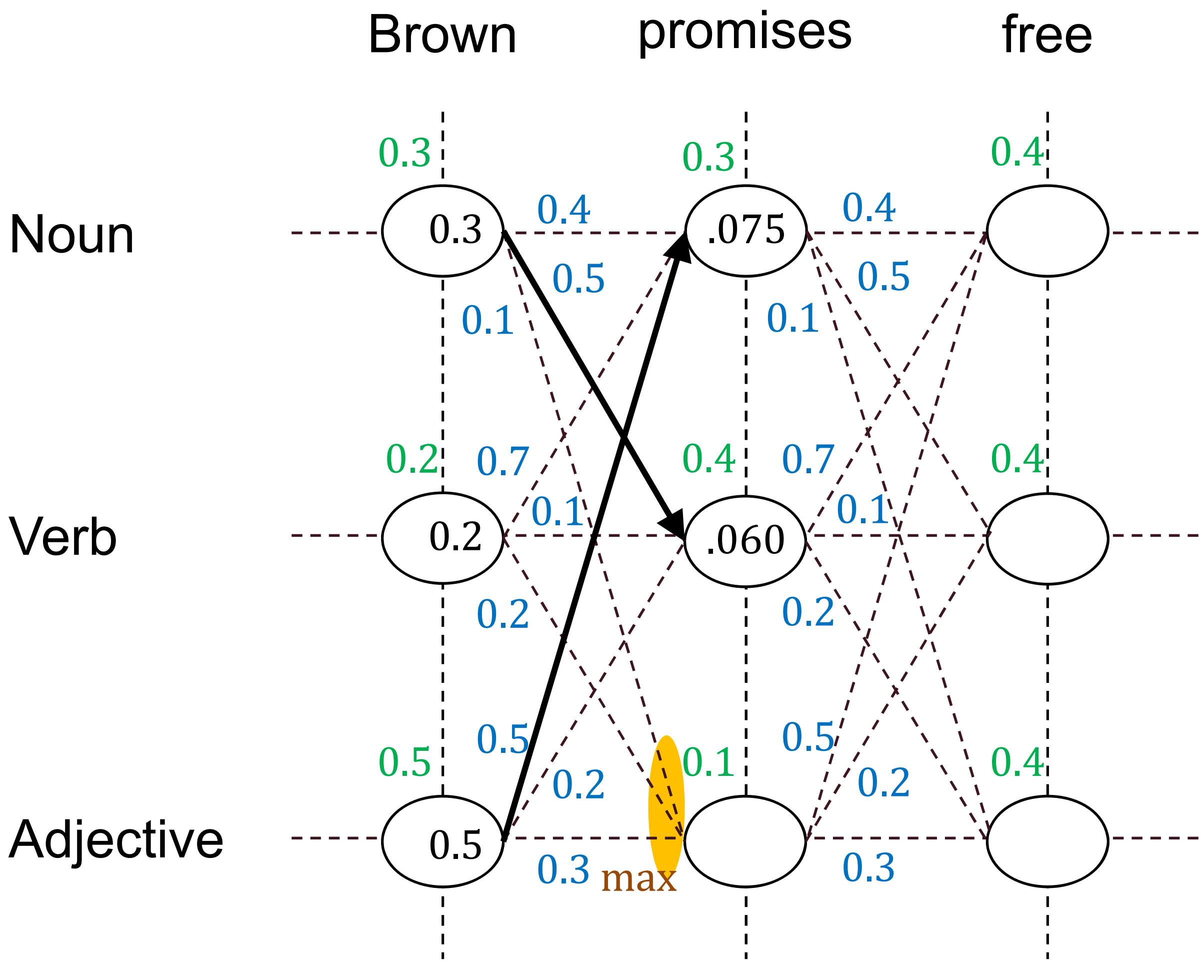


Noun - Verb
$0.3 \times 0.5 \times 0.4 = 0.060$

Verb - Verb
$0.2 \times 0.1 \times 0.4 = 0.008$

Adj - Verb
$0.5 \times 0.2 \times 0.4 = 0.040$

Viterbi (4/11) – Max route to adj at $t = 2$

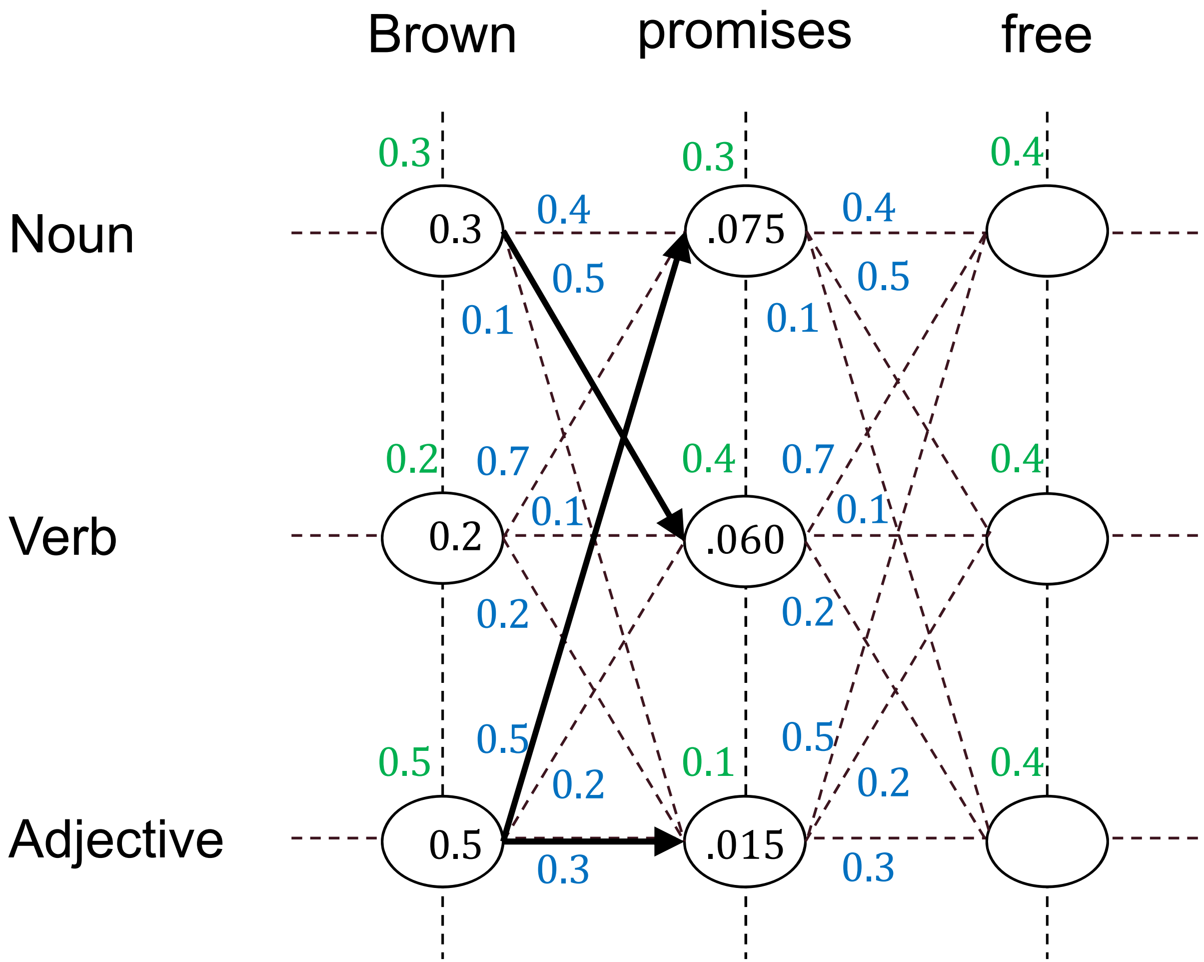


Noun - Adj
$0.3 \times 0.1 \times 0.1 = 0.003$

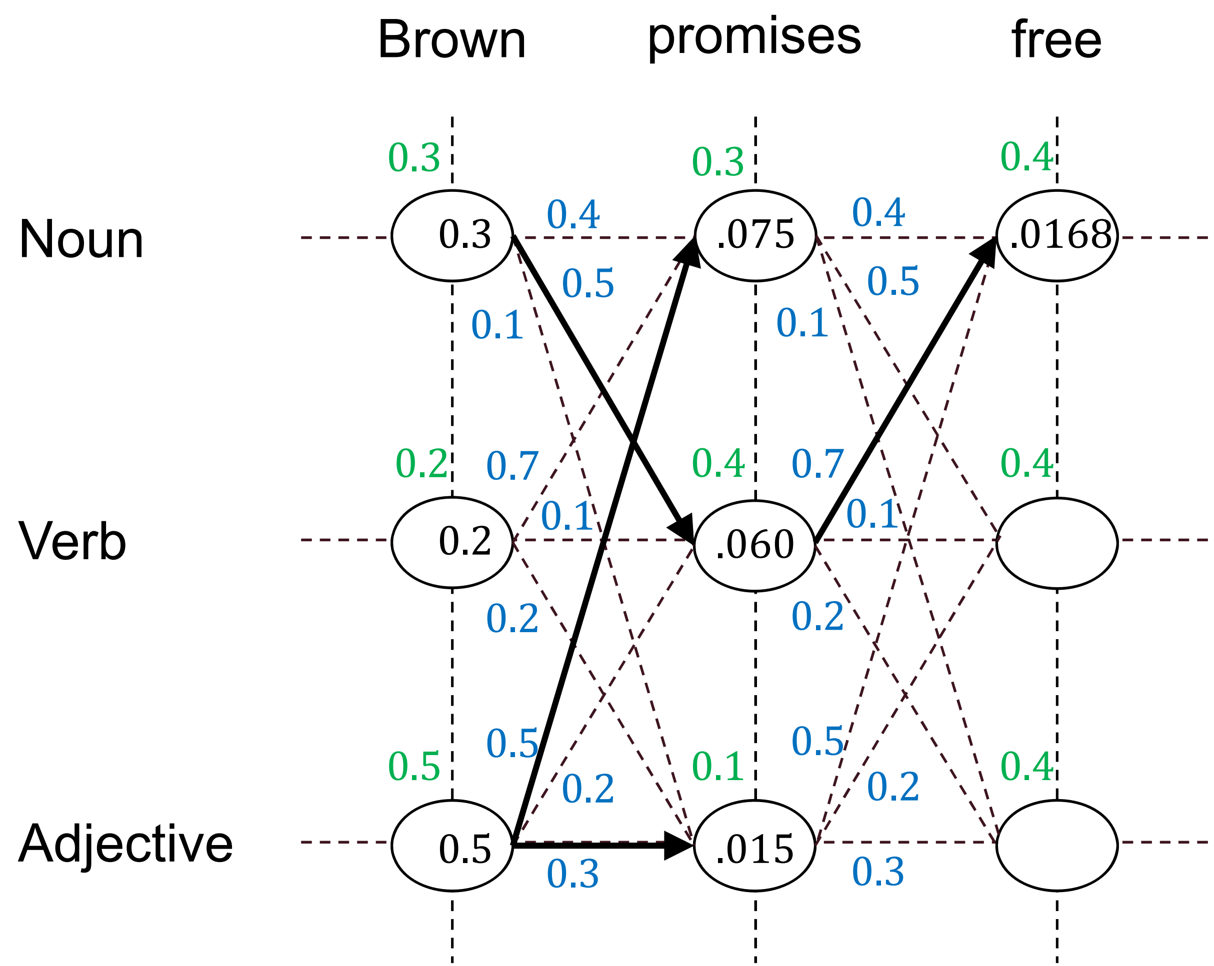
Verb - Adj
$0.2 \times 0.2 \times 0.1 = 0.004$

Adj - Adj
$0.5 \times 0.3 \times 0.1 = 0.015$

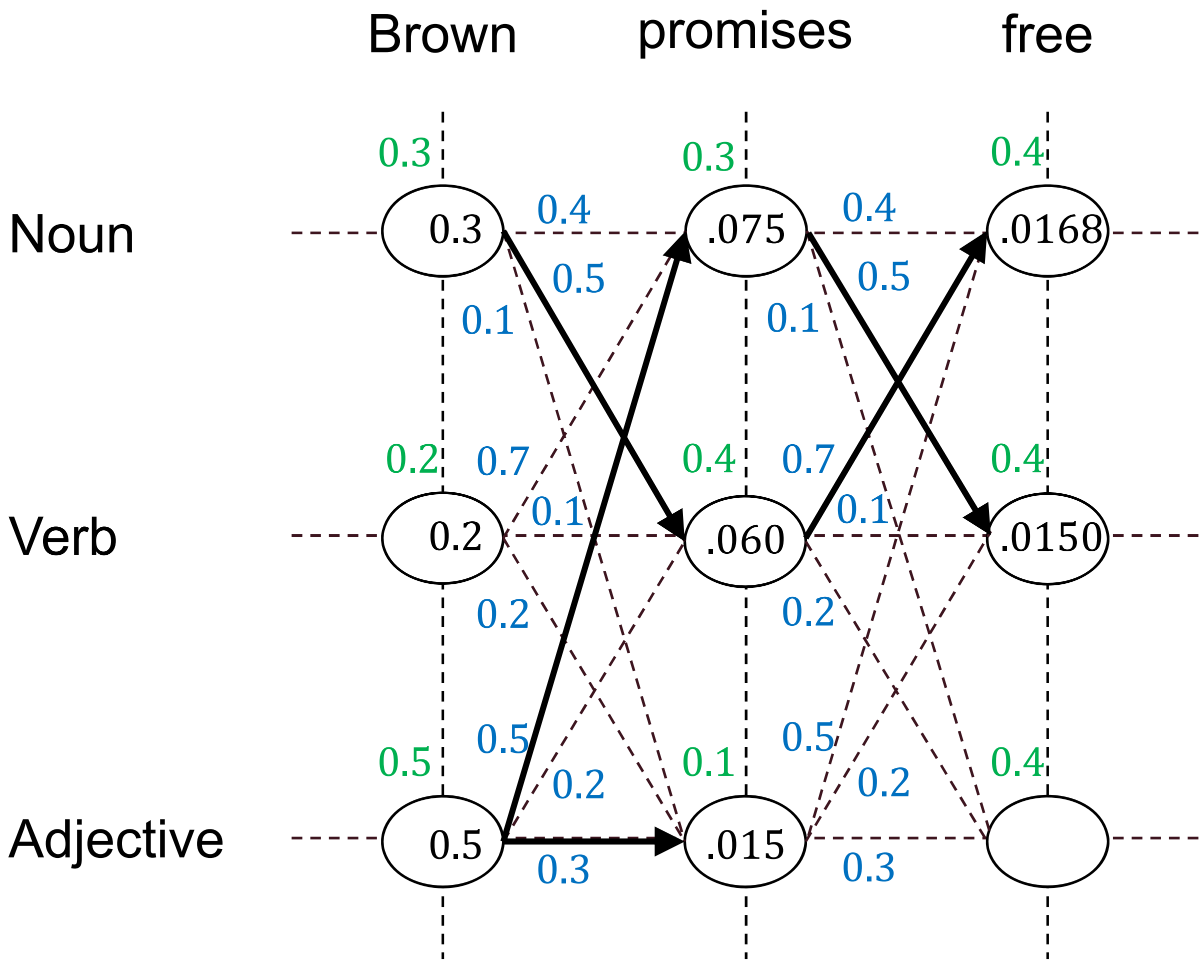
Viterbi (5/11) – Finished at $t = 2$



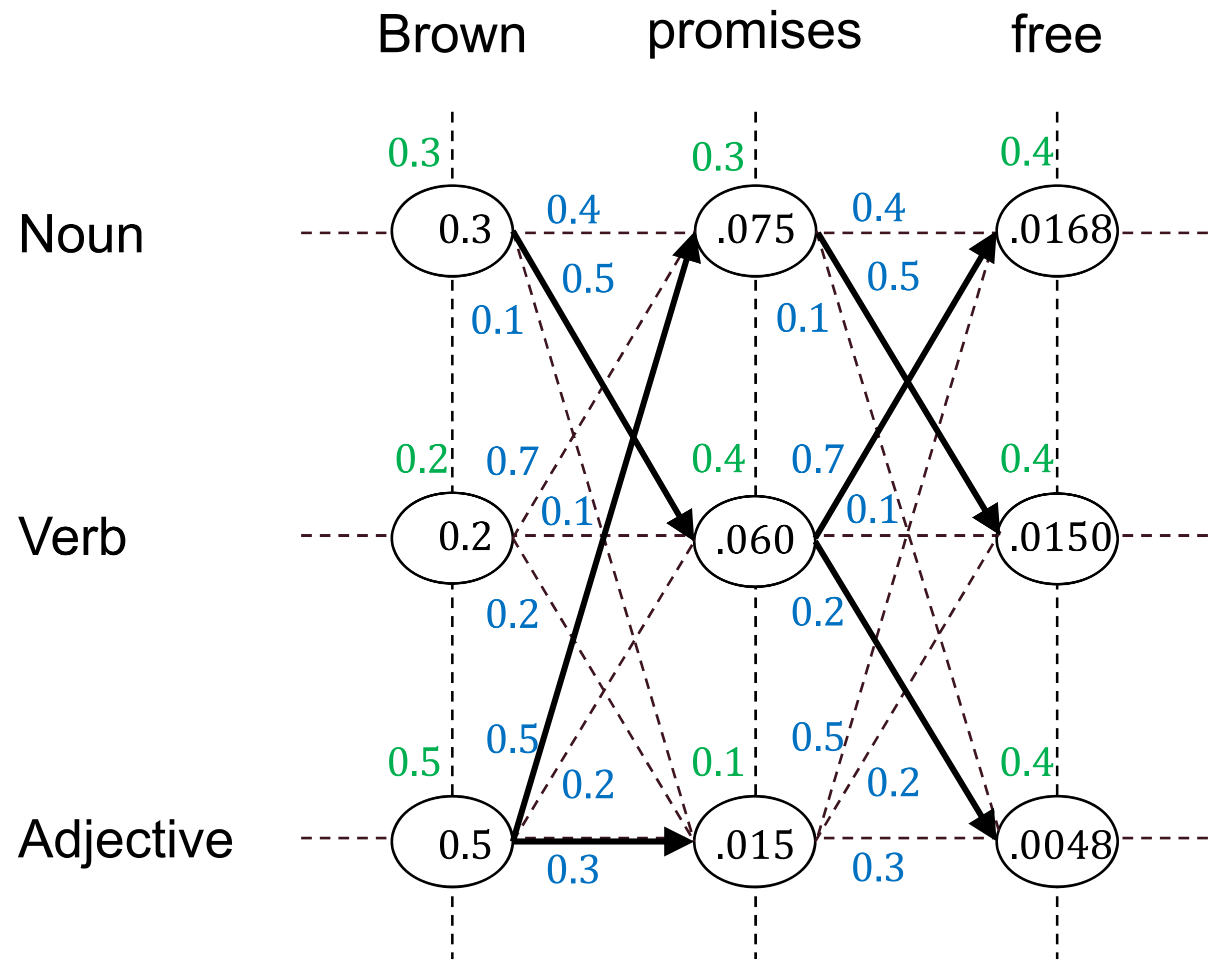
Viterbi (6/11) – Max route to noun at $t = 3$



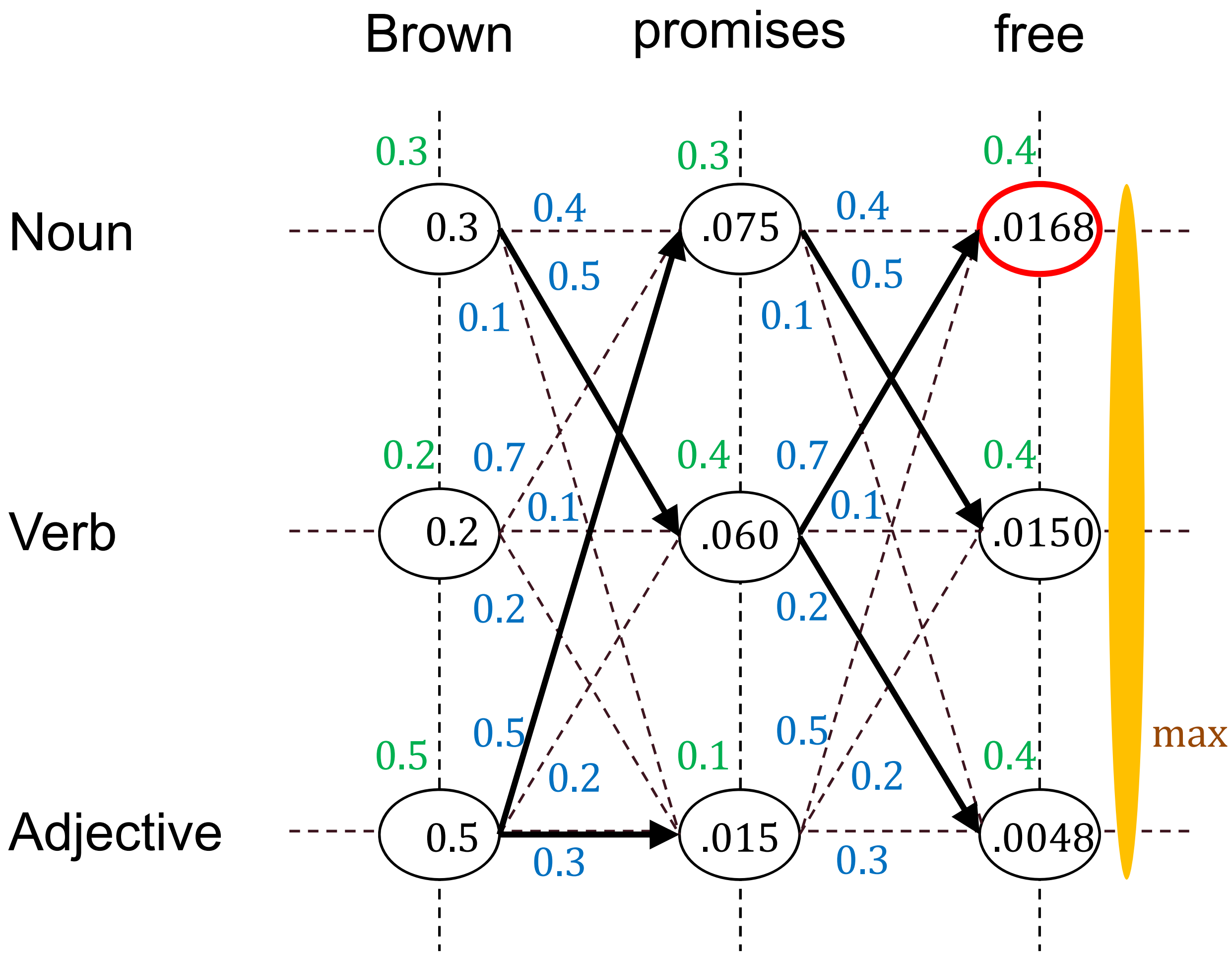
Viterbi (7/11) – Max route to verb at $t = 3$



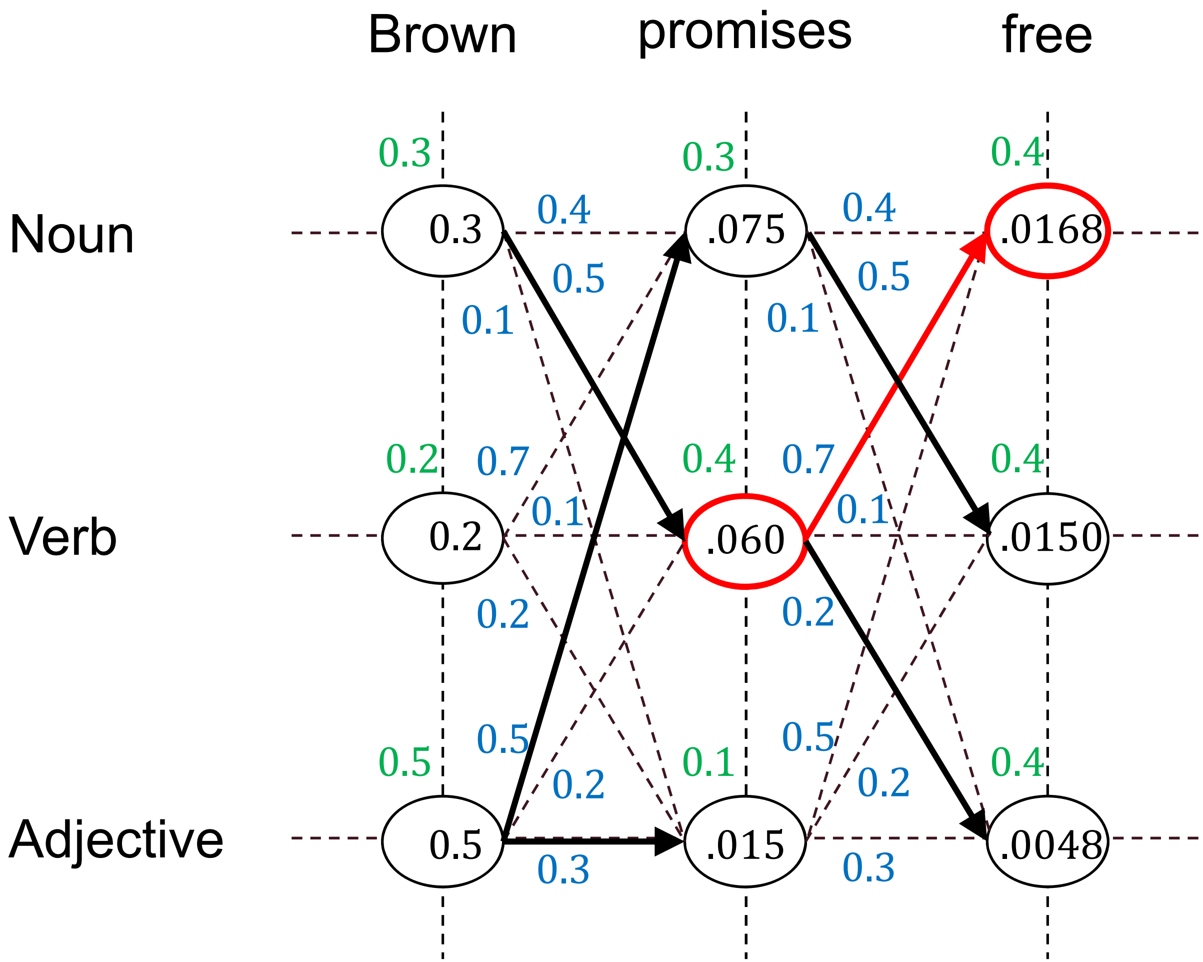
Viterbi (8/11) – Max route to adj at $t = 3$



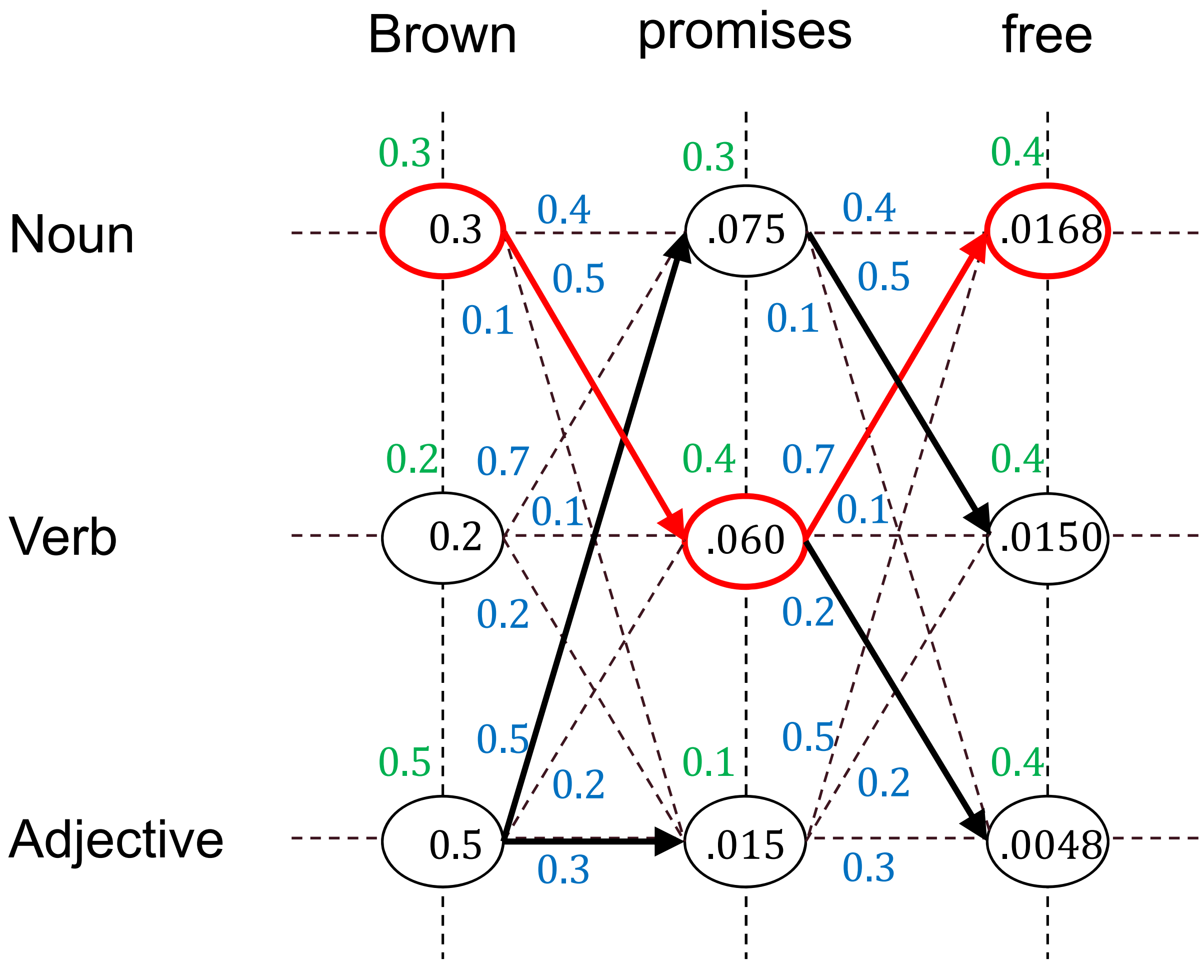
Viterbi (9/11) – Take the max at $t = 3$



Viterbi (10/11) – Trace back to $t = 2$



Viterbi (10/11) – Trace back to $t = 1$



$\max \phi(x, y) = 0.168$

$\hat{\boldsymbol{y}} =$ Noun Verb Noun

Practical considerations of HMM

- In practice, a smoothing method is used to avoid the zero-frequency problem (ゼロ頻度問題)
  - The model cannot predict a POS tag for an unknown word (that does not appear in a training set)
  - Smoothing: assign probability distributions for unknown words

- Use $\log P(x_t|y_t)$ and $\log P(y_t|y_{t-1})$ instead
  - Because products of probability values easily underflow on computer
  - Viterbi algorithm can be implemented with additions (not with multiplications)

Drawbacks of HMM-based POS tagging

- HMM cannot incorporate multiple types of associations between tokens and their POS tags
  - $P(x_t|y_t)$: association between tokens and their POS tags (only)
  - Other characteristics (e.g., prefixes, postfixes) may also be effective
  - $P(\text{supermassively}|RB)$ has nothing to do with $P(\text{lovely}|RB)$ even though the two tokens *lovely* and *supermassively* share postfix *–ly*
  - Weak generalization capability → data sparseness problem

- HMM is inflexible to memorize multiple spans of POS tags
  - at/IN all/DT
  - all/RB but/RB
  - all/RB right/JJ
  - all/PDT the/DT best/JJS

Memorizing these collocations would be much easier to model

Part-of-Speech Tagging using Structured Perceptron

(Not in the textbook)

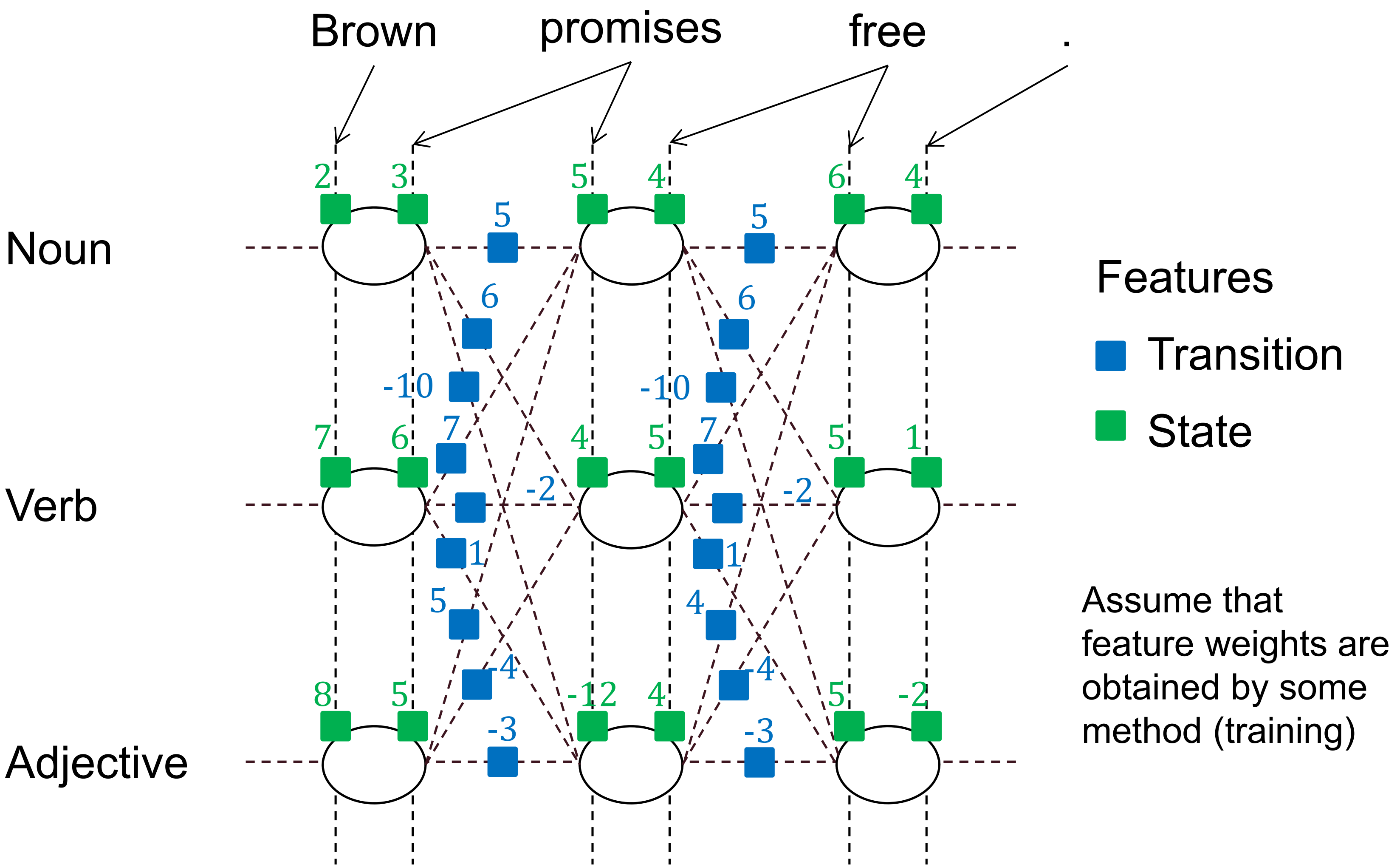
Structured perceptron (Collins, 2002)

- Natural extension to sequential labeling problem

- Replace probability distributions $P(y_t|y_{t-1})$ and $P(x_t|y_t)$ in HMM with features and their weights
  - $P(y_t|y_{t-1}) \rightarrow$ label bigram (transition) feature and its weight
  - $P(x_t|y_t) \rightarrow$ label unigram (state) feature(s) and their weight(s)

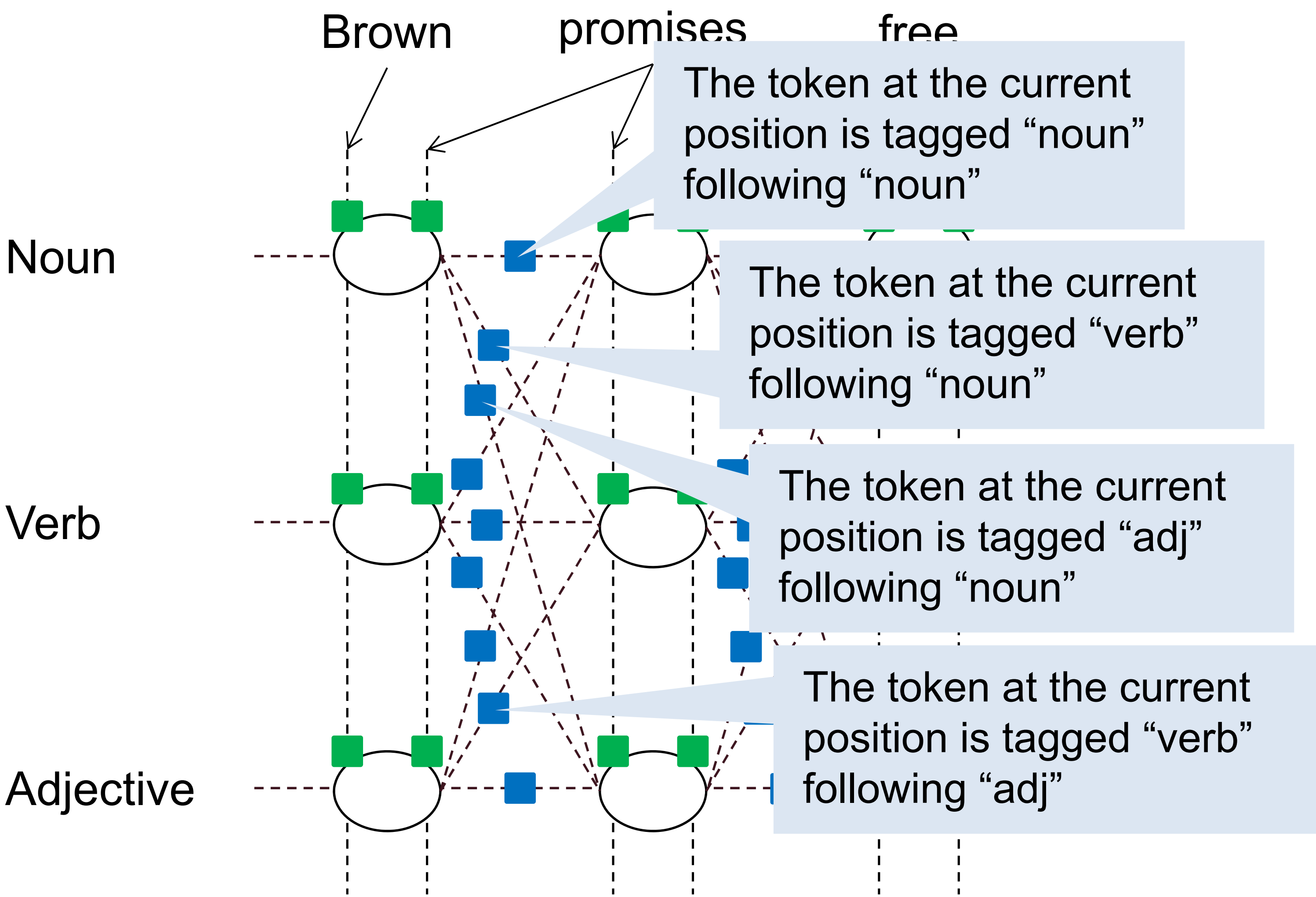
- Mathematical formula of structured perceptron is abstract
  - $\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, \boldsymbol{w} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$
  - The notation $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$ in this formula *implies many*
  - *Understand the graphical model represented by this formula first!*

Lattice representation of structured perceptron model (This is an example of feature design)

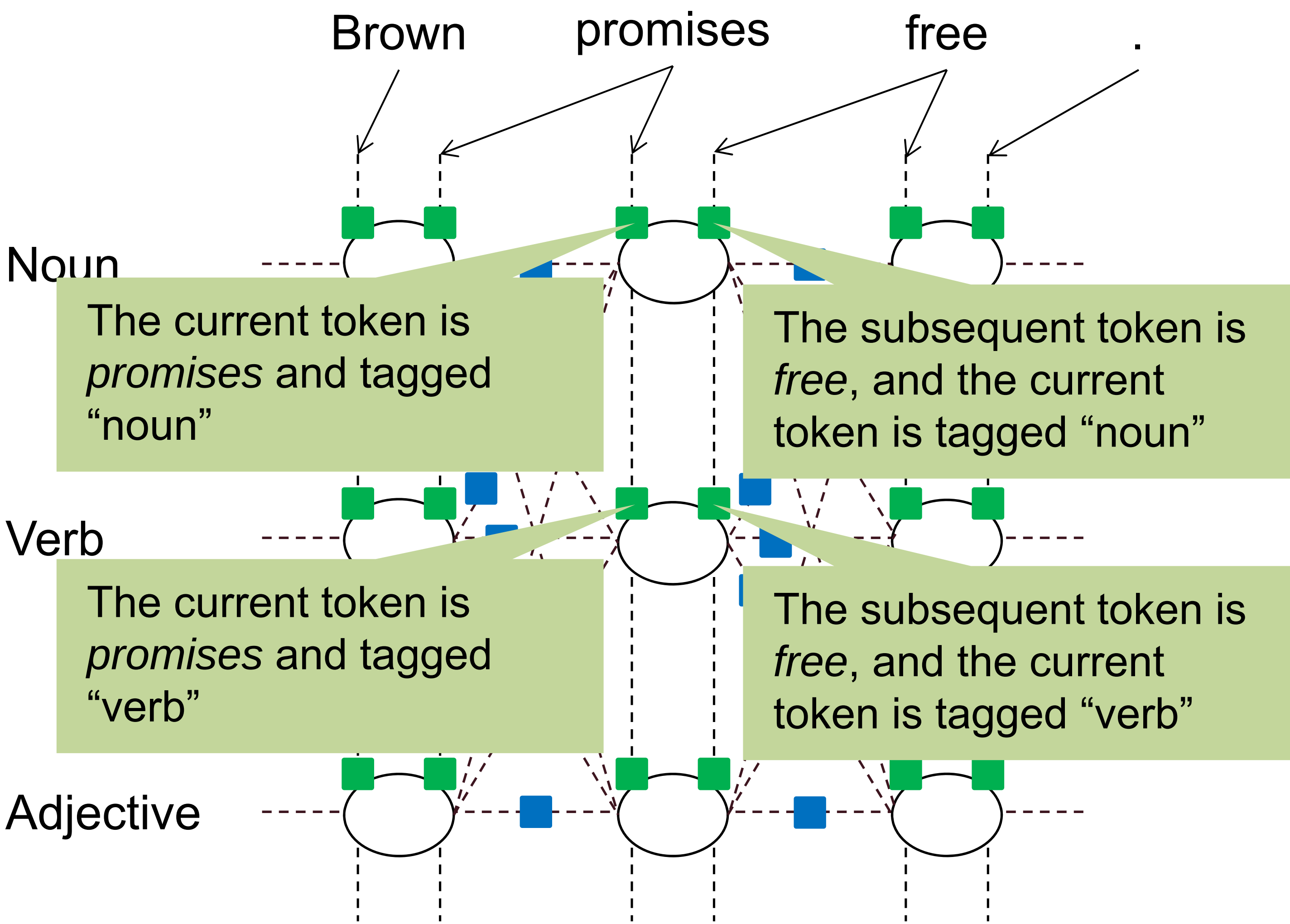


Assume that feature weights are obtained by some method (training)

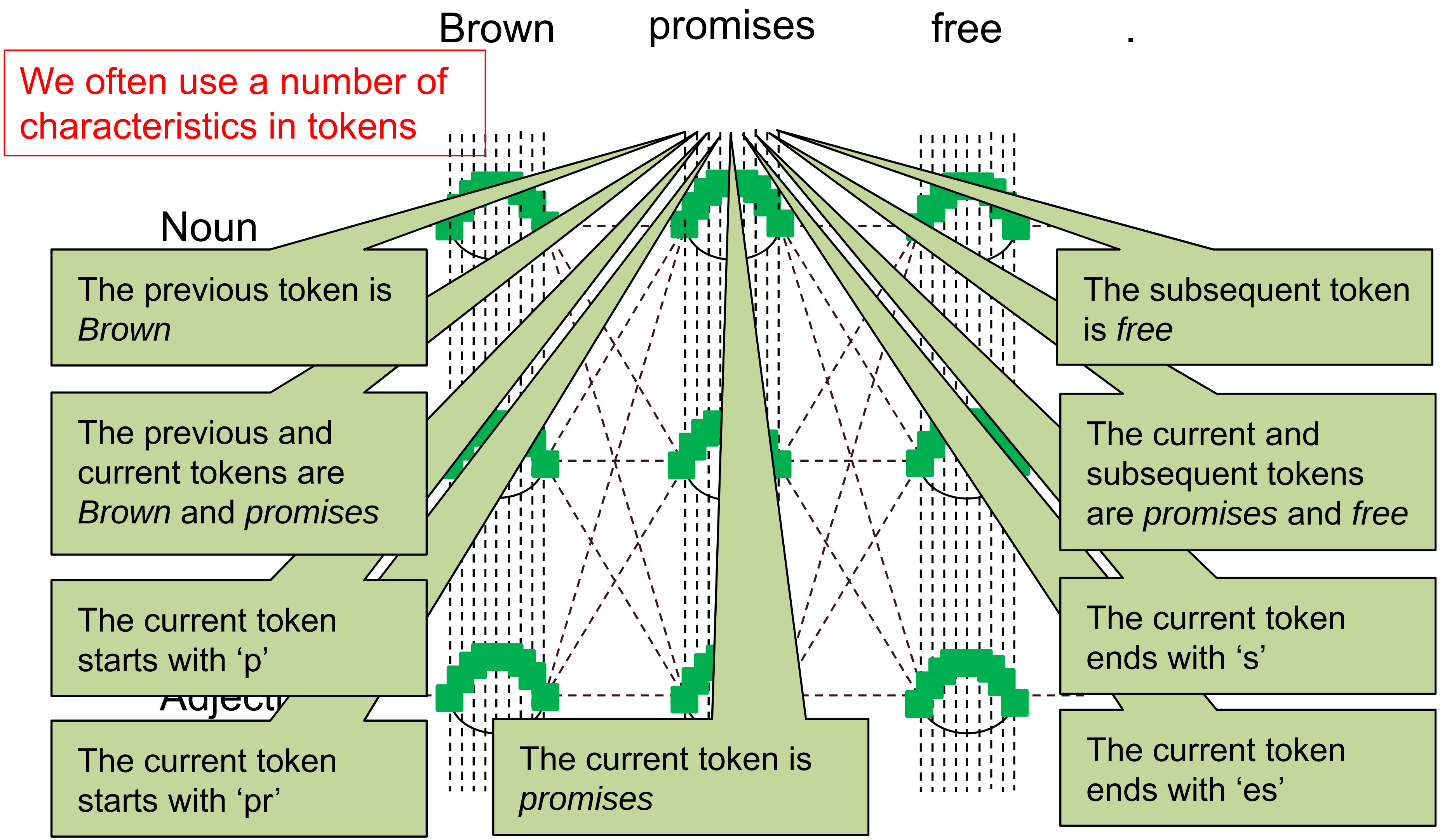
Transition (label-bigram) features



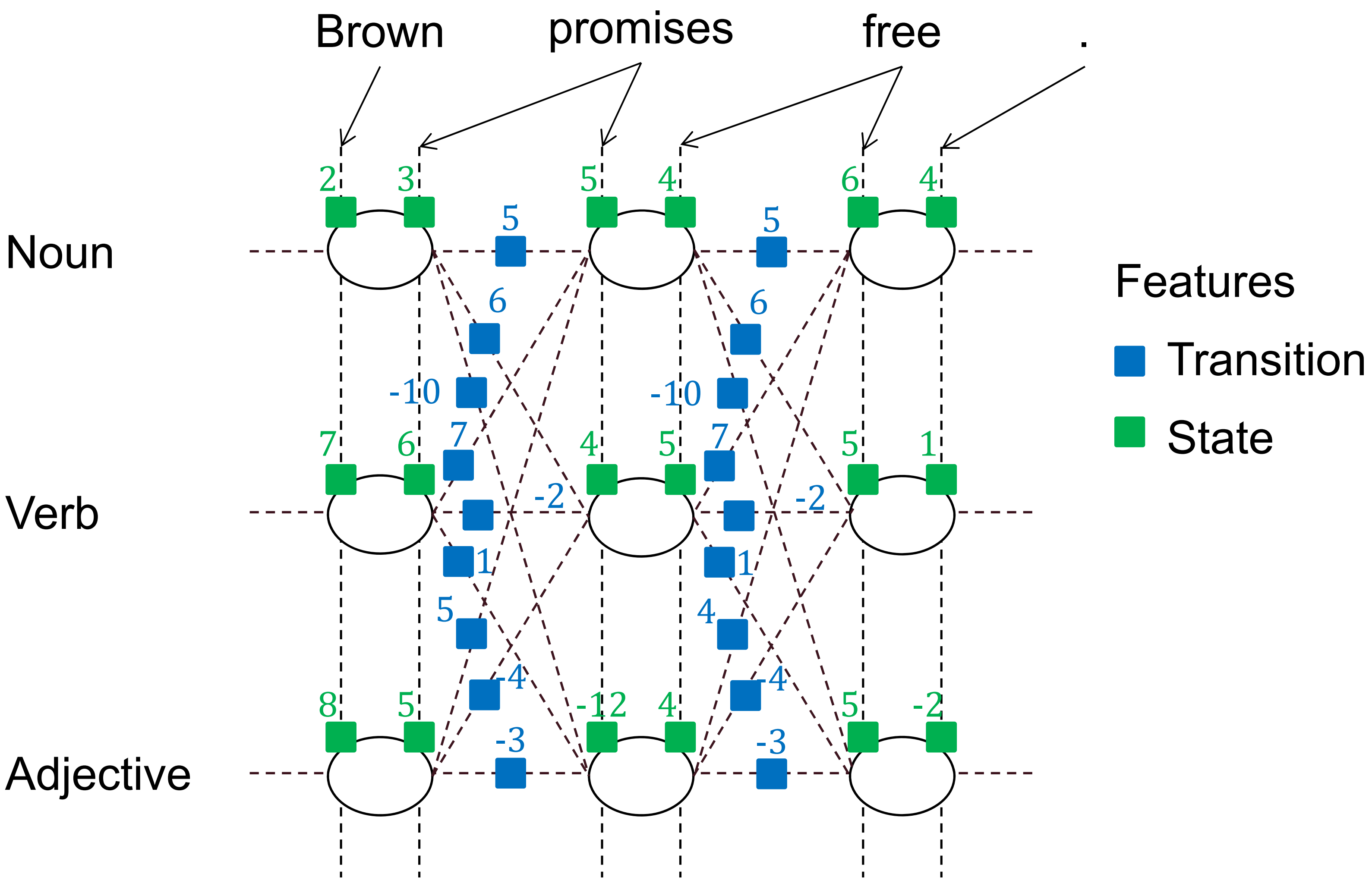
State (unigram) features



State features (more real design)



Going back to the lattice representation



Structured perceptron model

- *Path*: an assignment of part-of-speech tags

- The score of part-of-speech tags are defined by the sum of feature weights on the corresponding path on the lattice
  - $a(\boldsymbol{x}, \text{NN VB NN}) = (2+3) + 6 + (4+5) + 7 + (6+4) = 37$
  - $a(\boldsymbol{x}, \text{ADJ NN VB}) = (8+5) + 5 + (5+4) + 6 + (5+1) = 39$

- Part-of-speech tagging (inference):
  - To find the path that yields the maximum score $a(\boldsymbol{x}, \boldsymbol{y})$
  - $\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, a(\boldsymbol{x}, \boldsymbol{y})$
  - Use Viterbi algorithm to find $\hat{\boldsymbol{y}}$ (similarly to HMM)

Let's go back to the math

- Input: sequence of tokens $\boldsymbol{x} = (x_1\ x_2\ \ldots\ x_T)$
- Output: sequence of POS tags $\hat{\boldsymbol{y}} = (\widehat{y_1}\ \widehat{y_2}\ \ldots\ \widehat{y_T})$
- Mapping to global feature vector: $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$: $(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \mathcal{R}^m$

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t=1}^{T} \boxed{\{\boldsymbol{u}(x_t, y_t) + \boldsymbol{b}(y_{t-1}, y_t)\}}$$

Local feature vector (at t):
- Unigram feature vector
- Bigram feature vector

  - Each element of feature vector consists of a feature function, e.g.,
    - $u_{109}(x_t, y_t) = \{1\ (\text{if } x_t = \text{Brown } \textit{and } y_t = \text{Noun}); 0\ (\text{otherwise})\}$
    - $b_2(y_{t-1}, y_t) = \{1\ (\text{if } y_{t-1} = \text{Noun } \textit{and } y_t = \text{Verb}); 0\ (\text{otherwise})\}$
  - $\boldsymbol{u}(x_t, y_t)$ and $\boldsymbol{b}(y_{t-1}, y_t)$ are defined not to collide in feature space
    - $\underbrace{(\underline{\text{Used by } \boldsymbol{b}}\quad \underline{\text{Used by } \boldsymbol{u}})}_{m}$
- Using: weight vector $\boldsymbol{w} \in \mathcal{R}^m$
- Inference: $\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, a(\boldsymbol{x}, \boldsymbol{y})$, $a(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{w} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$

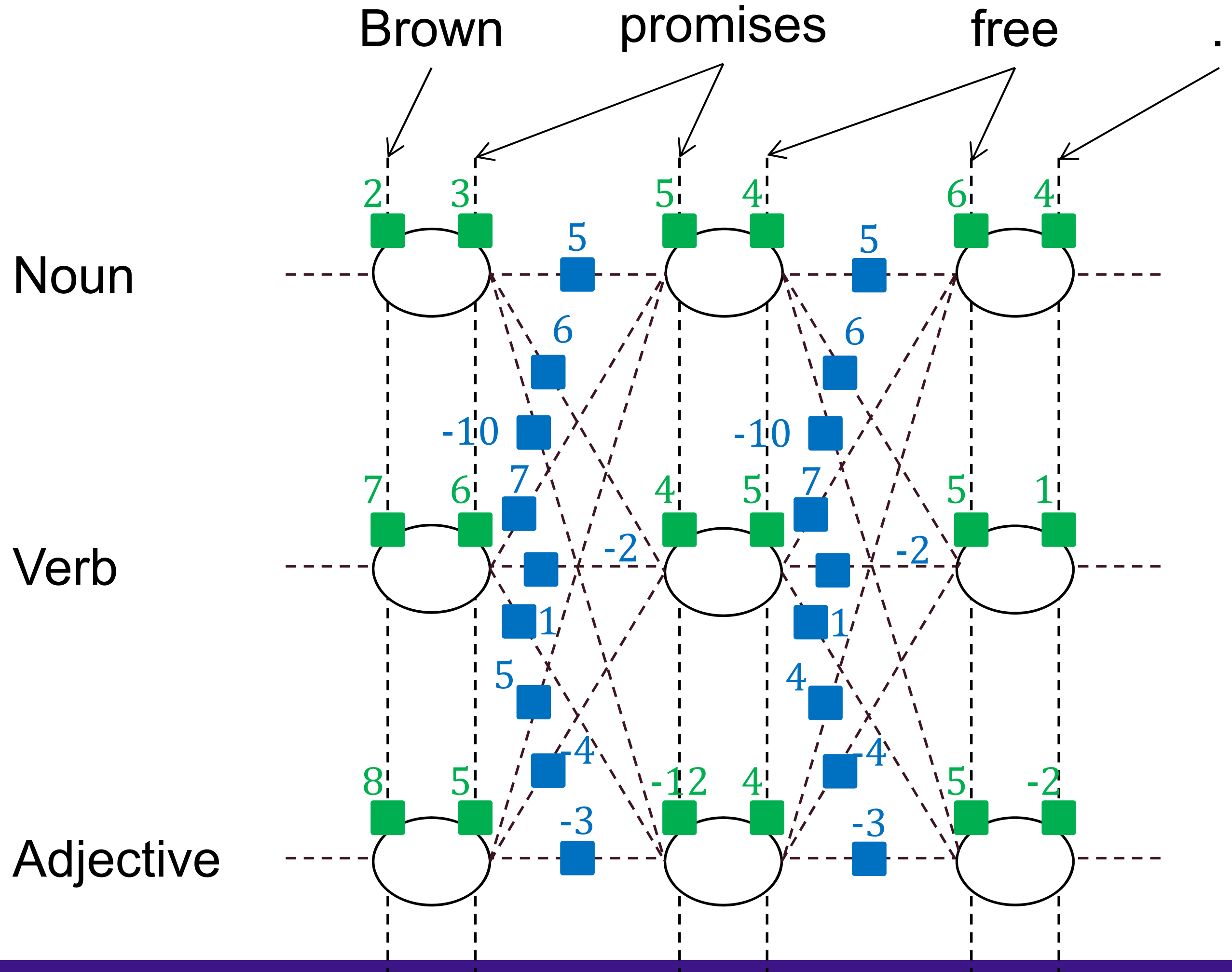
Training using perceptron

- We have a training data consisting of N instances:
  - $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$

1. $w_i = 0$ for all $i \in [1, m]$
2. Repeat:
3. $\quad (\boldsymbol{x}_n, \boldsymbol{y}_n) \longleftarrow$ Random sample from the training data D
4. $\quad \hat{\boldsymbol{y}} \longleftarrow \underset{\boldsymbol{y}}{\operatorname{argmax}}\, \boldsymbol{w} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$
5. $\quad$ if $\hat{\boldsymbol{y}} \neq \boldsymbol{y}_n$ then:
6. $\qquad \boldsymbol{w} \longleftarrow \boldsymbol{w} + \boldsymbol{F}(\boldsymbol{x}_n, \boldsymbol{y}_n) - \boldsymbol{F}(\boldsymbol{x}_n, \hat{\boldsymbol{y}})$
7. Until convergence (e.g., until no instance updates $\boldsymbol{w}$)

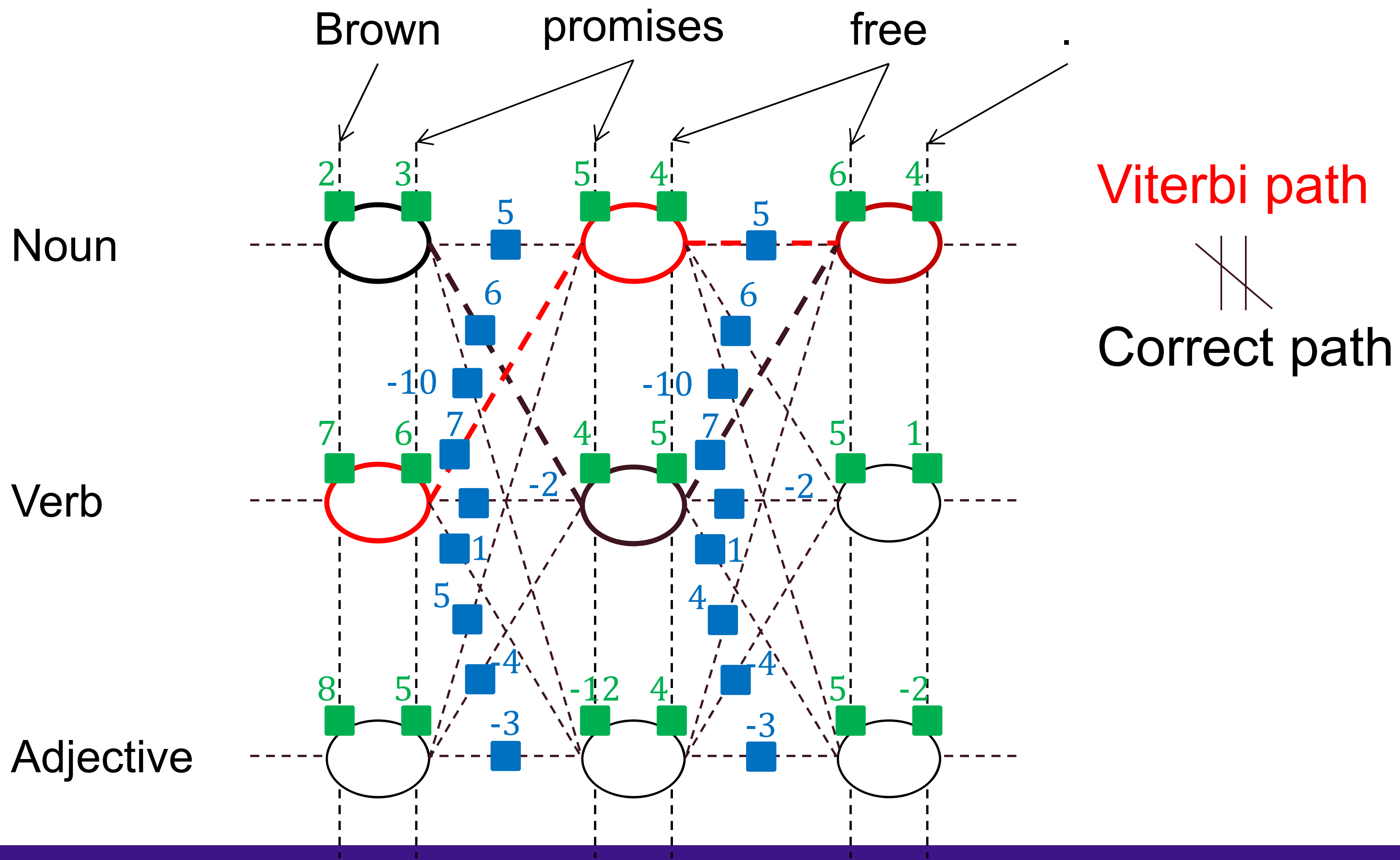
Perceptron update in the lattice graph (1/3)

- $(\boldsymbol{x}_n, \boldsymbol{y}_n) = (\text{Brown promises change}, \text{Noun Verb Noun})$



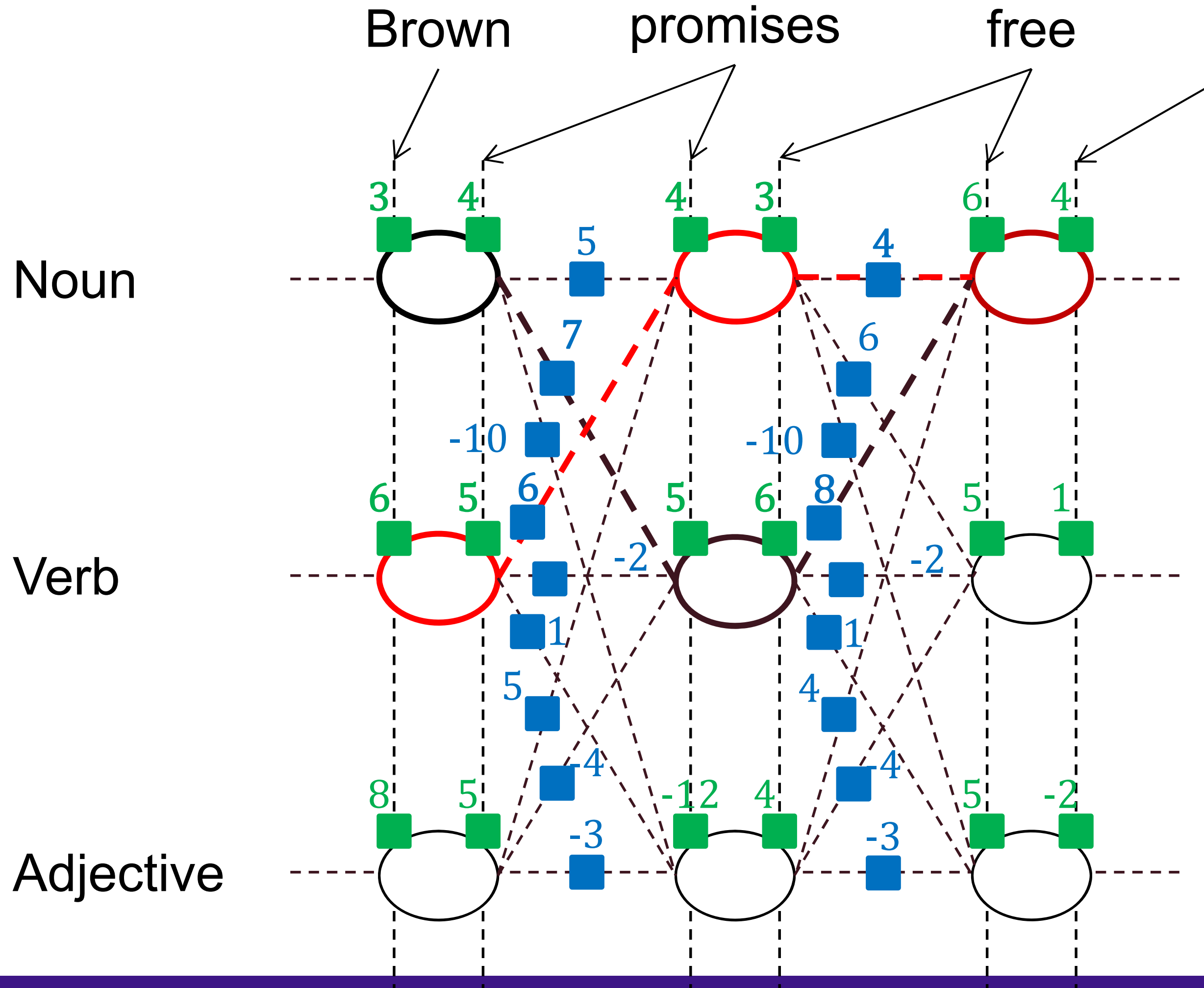
Perceptron update in the lattice graph (2/3)

- $(\boldsymbol{x}_n, \boldsymbol{y}_n) = (\text{Brown promises change, Noun Verb Noun})$



Perceptron update in the lattice graph (3/3)

- $(\boldsymbol{x}_n, \boldsymbol{y}_n) = (\text{Brown promises change, Noun Verb Noun})$



Viterbi path

$\neq$

Correct path

Update:

Increment weights on the correct path

Decrement weights on the Viterbi path

Notes on structured perceptron

- This algorithm surprisingly works well despite its simplicity

- The same practical considerations of ‘unstructured’ version apply to the structured version

Conditional Random Fields (CRFs) (Lafferty+ 2001)

- The same graphical model as structured perceptron

- Conditional probability is defined,

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{w} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})\right)}{\boxed{\sum_{\boldsymbol{y}} \exp\left((\boldsymbol{w} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})\right)}}$$

Normalized by the sum of exp'd scores of all possible paths in the lattice

- The same inference algorithm (Viterbi)

- Training with Stochastic Gradient Descent has the same update rule as logistic regression
  - Updating feature weights by the amount of error
  - Requires forward-backward (alpha-beta) algorithm, a kind of dynamic programing, for computing the partition factor (分配関数) and marginal probabilities (周辺確率) (of feature occurrences)

Other tasks

13.5 (Partial Parsing)

Sequential labeling problem and NLP

- *Many NLP problems can be formalized as sequential labeling!*

In March 2005, the New York Times acquired About, Inc.

In	March	2005,	the	New	York	Times	acquired	About,	Inc.
IN	NNP	CD	DT	NNP	NNP	NNP	VBD	NNP	NNP
	NP		NP				VP	NP	
	TEMP			ORG				ORG	

IOB2 notation for representing segments

- *Many NLP problems can be formalized as sequential labeling!*

In March 2005, the New York Times acquired About, Inc.

IN	NNP	CD	DT	NNP	NNP	NNP	VBD	NNP	NNP
O	B-NP	I-NP	B-NP	I-NP	I-NP	I-NP	B-VP	B-NP	I-NP
O	B-TEMP	I-TEMP	O	B-ORG	I-ORG	I-ORG	O	B-ORG	I-ORG

- *Segments can be represented by IOB2 notation*

Implementations

Implementations for sequential labeling

- CRF++: http://crfpp.sourceforge.net/
  - C++ implementation
- MALLET: http://mallet.cs.umass.edu/
  - Java implementation; this software includes other ML algorithms
- CRFsuite: http://www.chokkan.org/software/crfsuite/
  - C implementation

References

- Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. EMNLP 2002.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. ICML 2001.
- Beatrice Santorini. 1990. Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision), Technical Report, University of Pennsylvania.

Appendix

Python implementation of training (hmm.py) (1/2)

```
import collections

def to_probdist(M): # Convert frequency table to probability distribution
    for row, vec in M.iteritems():
        n = sum(vec.itervalues())
        for x, p in vec.iteritems():
            vec[x] /= n

def train(D):
    S = collections.defaultdict(lambda: collections.defaultdict(float))
    T = collections.defaultdict(lambda: collections.defaultdict(float))

    for seq in D:
        prev = None
        for token, label in seq:
            S[label][token] += 1                # Count up word emissions
            if prev is not None:
                T[prev][label] += 1             # Count up label transitions
            prev = label

    to_probdist(S)
    to_probdist(T)
    return S, T
```

Python implementation of training (hmm.py) (2/2)

```python
D = (                    # Training data
    (                    # The 1st sentence
        ("The", 'DT'),
        ("growing", 'VBG'),
        ("crowd", 'NN'),
        ("of", 'IN'),
        ("Japanese", 'JJ'),
        ("investors", 'NNS'),
        ("buying", 'VBG'),
        ("up", 'RP'),
        ("foreign", 'JJ'),
        ("companies", 'NNS'),
        ("are", 'VBP'),
        ("n't", "RB"),
        ("all", "RB"),
        ("strait-laced", "JJ"),
        ("businessmen", "NNS"),
        ("in", "IN"),
        ("dark", "JJ"),
        ("suits", "NNS"),
        (".", "."),
    ),
    (                    # The 2nd sentence
        ("Yasumichi", "NNP"),
        ("Morishita", "NNP"),
        (",", ","),
        ("whose", "WP$"),
        ("art", "NN"),
        ("gallery", "NN"),
        ("last", "JJ"),
        ("month", "NN"),
        ("became", "VBD"),
    ),
)

S, T = train(D)
print S                  # Print emissions
print T                  # Print transitions
```

Obtained model (2/2)

- Transition probability distribution
  - VBG: {'RP': 0.5, 'NN': 0.5}
  - RB: {'RB': 0.5, 'JJ': 0.5}
  - NN: {'IN': 0.25, 'NN': 0.25, 'JJ': 0.25, 'VBD': 0.25}
  - ,: {'WP$': 1.0}
  - VBP: {'RB': 1.0}
  - JJ: {'NNS': 0.8, 'NN': 0.2}
  - IN: {'JJ': 1.0}
  - WP$: {'NN': 1.0}
  - RP: {'JJ': 1.0}
  - DT: {'VBG': 1.0}
  - NNS: {'VBP': 0.25, 'VBG': 0.25, 'IN': 0.25, '.': 0.25}
  - NNP: {',': 0.5, 'NNP': 0.5}