Natural Language Processing: CIS 410/510

Sequence Labeling

Instructor: Thien Huu Nguyen

Based on slides from: Ralph Grishman, David Bamman, Dan Jurasky, Chris Manning and others

Parts of Speech (POS)

- Grammar is stated in terms of parts of speech ('preterminals'):
  - classes of words sharing syntactic properties:

noun

verb

adjective

…

Parts of Speech (POS)

- The distributional hypothesis: Words that appear in similar contexts have similar representations (and similar meanings)

- Substitution test for POS: if a word is replaced by another word, does the sentence remain grammatical?

He noticed the	*elephant*	*before anybody else*
	dog	
	cat	
	point	
	features	
	**what*	
	**and*	

Substitution test

- These can often be too strict; some contexts admit substitutability for some pairs but not others.

He noticed the *elephant* *before anybody else*

**Sandy*

Both nouns
but common vs. proper

*He *arrived the* *elephant* *before anybody else*

Both verbs
but transitive vs. intransitive

Parts of Speech (POS)

Nouns	People, places, things, actions-made-nouns ("I like swimming"). Inflected for singular/plural
Verbs	Actions, processes. Inflected for tense, aspect, number, person
Adjectives	Properties, qualities. Usually modify nouns
Adverbs	Qualify the manner of verbs ("She ran downhill extremely quickly yesterday")
Determiner	Mark the beginning of a noun phrase ("a dog")
Pronouns	Refer to a noun phrase (he, she, it)
Prepositions	Indicate spatial/temporal relationships (on the table)
Conjunctions	Conjoin two phrases, clauses, sentences (and, or)

POS Tag Sets (Categories)

Most influential tag sets were those defined for projects to produce large POS-annotated corpora:

- Brown corpus
  - 1 million words from variety of genres
  - 87 tags
- UPenn Tree Bank
  - initially 1 million words of Wall Street Journal
  - later retagged Brown
  - first POS tags, then full parses
  - 45 tags (some distinctions captured in parses)

Penn Treebank POS Tags

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	“to”	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential ‘there’	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*’s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one’s*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Verbs

Tag	Description	Examples
VB	base form (found in imperatives, infinities and subjunctives)	• Just do it • You should do it • He wants to do it
VBD	past tense	• He ate the food
VBG	present participle (Verb forms in the gerund or present participle; generally end in -ing)	• He was going to the store • She is implementing the algorithm
VBN	past participle	• The apple was eaten • He had expected to go
VBP	present (non 3rd-sing)	• I am the food • You are tall • We are tall • They do the job
VBZ	present (3rd-sing)	• She is tall • He likes ice cream
MD	modal verbs (All verbs that don't take -s ending in third-person singular present)	• can, could, dare, may, might, must, ought, shall, should, will, would

```
4057 will/md
2973 would/md
1483 could/md
1233 can/md
1066 may/md
 598 should/md
 459 might/md
 332 must/md
 326 wo/md
 246 ca/md
```

Nouns

Tag	Description	Examples
NN	non-proper, singular or mass	the company
NNS	non-proper, plural	the companies
NNP	proper, singular	Carolina
NNPS	proper, plural	Carolinas

RP (particle)

- Used in combination with a verb
  - She turned the paper over
- verb + particle = phrasal verb, often non-compositional
  - turn down, rule out, find out, go on

```
774 up/rp
487 out/rp
301 off/rp
209 down/rp
124 in/rp
 98 over/rp
 81 on/rp
 72 back/rp
 46 around/rp
 25 away/rp
```

DT and PDT

- DT (Articles)
  - Articles (a, the, every, no)
  - Indefinite determines (another, any, some, each)
  - That, these, this, those when preceding noun
  - All, both when not preceding another determiner or possessive pronoun

- PDT (Predeterminer)
  - Determiner-like words that precede an article or possessive pronoun
    - all his marbles
    - both the girls
    - such a good time

```
65548 the/dt
26970 a/dt
 4405 an/dt
 3115 this/dt
 2117 some/dt
 2102 that/dt
 1274 all/dt
 1085 any/dt
  953 no/dt
  778 those/dt
```

```
263 all/pdt
114 such/pdt
 84 half/pdt
 24 both/pdt
  7 quite/pdt
  2 many/pdt
  1 nary/pdt
```

PRP and PRP$

- PRP (personal pronoun)
  - Personal pronouns (I, me, you, he, him, it, etc.)
  - Reflective pronouns (ending in -self): himself, herself
  - Nominal possessive pronouns: mine, yours, hers

```
7854 it/prp
4601 he/prp
3260 they/prp
2323 his/prp$
1792 we/prp
1584 i/prp
1001 you/prp
 874 them/prp
 694 she/prp
 438 him/prp
```

- PRP$ (possessive pronouns)
  - Adjectival possessive forms: my, their, its, his, her

```
5013 its/prp$
2364 their/prp$
2323 his/prp$
 521 our/prp$
 430 her/prp$
 328 my/prp$
 269 your/prp$
```

Adjectives

- JJ (Adjectives)
  - General adjectives (happy person, new house)
  - Ordinal numbers (fourth cat)
- JJR (Comparative adjectives)
  - Adjectives with a comparative ending -er and comparative meaning (happier person)
  - More and less (when used as adjectives) (more mail)
- JJS (Superlative adjectives)
  - Adjectives with a superlative ending -est and superlative meaning (happiest person)
  - Most and least (when used as adjectives) (most mail)

```
2002 other/jj
1925 new/jj
1563 last/jj
1174 many/jj
1142 such/jj
1058 first/jj
 824 major/jj
 715 federal/jj
 698 next/jj
 644 financial/jj

1498 more/jjr
 518 higher/jjr
 432 lower/jjr
 285 less/jjr
 158 better/jjr
 136 smaller/jjr
 122 earlier/jjr
 112 greater/jjr
  93 larger/jjr
  75 bigger/jjr

 695 most/jjs
 428 least/jjs
 315 largest/jjs
 299 latest/jjs
 209 biggest/jjs
 194 best/jjs
  76 highest/jjs
  63 worst/jjs
  31 lowest/jjs
  30 greatest/jjs
```

Adverbs

- RB (Adverbs)
  - Most words that end in –ly (highly, heavily)
  - Degree words (quite, too, very)
  - Negative markers (not, n’t, never)
- RBR (Comparative adverbs)
  - Adverbs with a comparative ending -er and comparative meaning
  - More/less
- RBS (Superlative adverbs)
  - Adverbs with a superlative ending -est and superlative meaning.
  - Most/least

```
4410 n't/rb
2071 also/rb
1858 not/rb
1109 now/rb
1070 only/rb
1027 as/rb
 961 even/rb
 839 so/rb
 810 about/rb
 804 still/rb

1121 more/rbr
 516 earlier/rbr
 192 less/rbr
  88 further/rbr
  82 lower/rbr
  75 better/rbr
  65 higher/rbr
  57 longer/rbr
  53 later/rbr
  34 faster/rbr

549 most/rbs
 21 best/rbs
  9 least/rbs
  8 hardest/rbs
  2 most/rbs|jjs
  1 worst/rbs
  1 rbs/nnp
  1 highest/rbs
  1 earliest/rbs
```

IN and CC

- IN (preposition, subordinating conjunction)
  - All prepositions (except to) and subordinating conjunctions
    - He jumped on the table because he was excited
- CC (coordinating conjunction)
  - And, but, not, or
  - Math operators (plus, minor, less, times)
  - For (meaning "because")
    - he asked to be transferred, for he was unhappy

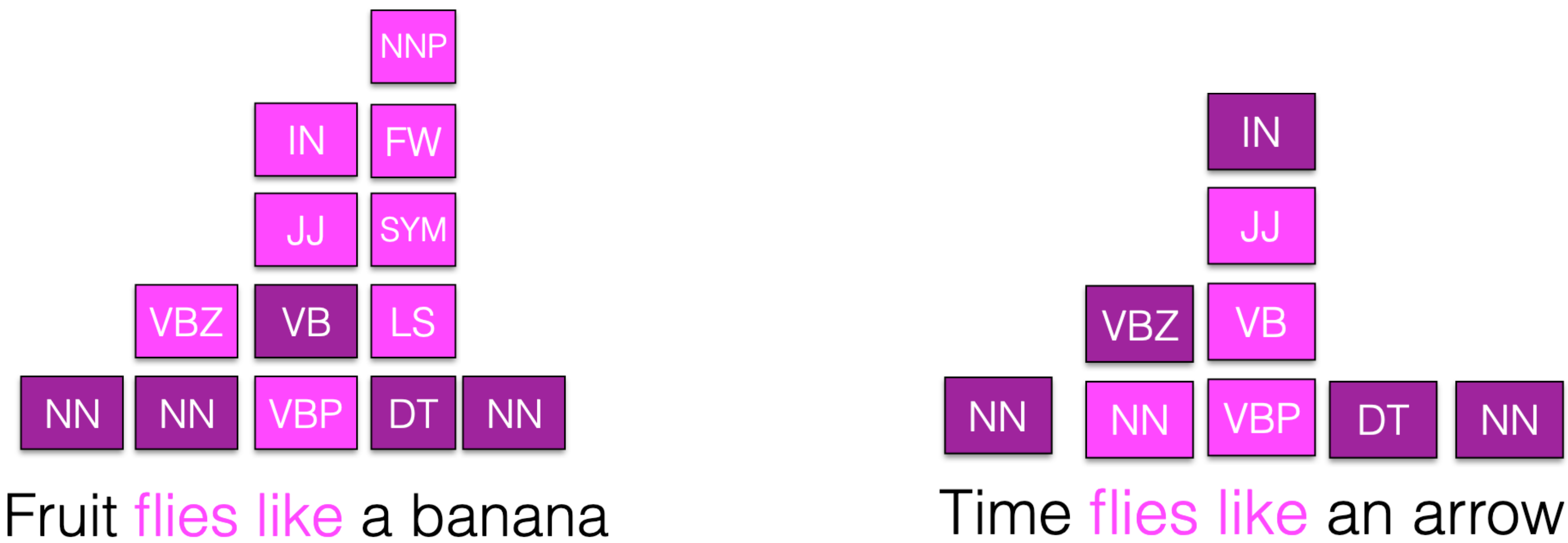
```
31111 of/in
22967 in/in
11425 for/in
 7181 on/in
 6684 that/in
 6399 at/in
 6229 by/in
 5940 from/in
 5874 with/in
 5239 as/in
```

```
22362 and/cc
 4604 but/cc
 3436 or/cc
 1410 &/cc
   94 nor/cc
   68 either/cc
   53 yet/cc
   53 plus/cc
   37 both/cc
   32 neither/cc
```

The POS tagging task

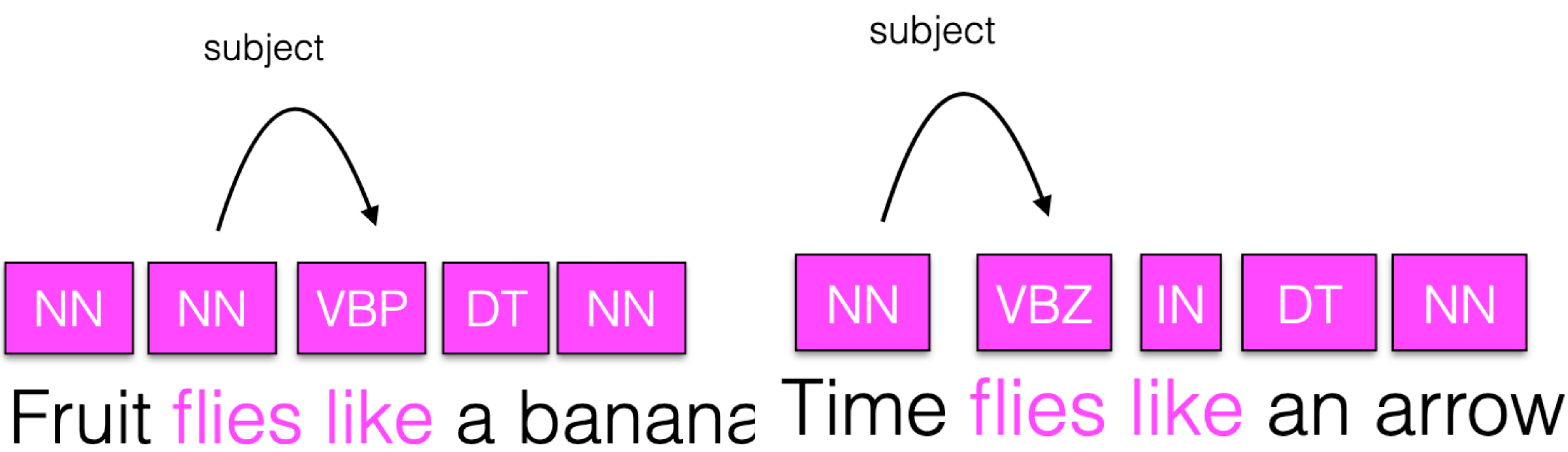
Task: assigning a POS to each word

- not trivial: many words have several tags
- dictionary only lists possible POS, independent of context



Why tag?

- POS tagging can help parsing by reducing ambiguity
- Can resolve some pronunciation ambiguities for text-to-speech (“desert” – noun: /ˈdɛzərt/, verb: /dɪˈzɜrt/)
- Can resolve some semantic ambiguities



Some tricky cases

- JJ or VBN
  - If it is gradable (can insert "very") = JJ
    - He was very surprised JJ
  - If can be followed by a "by" phrase = VBN. If that conflicts with #1 above, then = JJ
    - He was invited by some friends of her VBN
    - He was very surprised by her remarks JJ
- JJ or NP/NPS
  - Proper names can be adjectives or nouns
    - French cuisine is delicious JJ
    - The French tend to be inspired cooks NNPS

Some tricky cases

- NN or VBG
  - Only nouns can be modified by adjectives; only gerunds can be modified by adverbs
    - Good cooking is something to enjoy
    - Cooking well is a useful skill

NN
VBG

- IN or RP
  - If it can precede or follow the noun phrase = RP
    - She told off her friends
    - She told her friends off
  - If it must precede the noun phrase = IN
    - She stepped off the train
    - *She stepped the train off

Exercise [SLP2]

- Find the tagging errors in the following sentences:

I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NN

Does/VBZ this/DT flight/NN serve/VB dinner/NNS

I/PRP have/VB a/DT friend/NN living/VBG /in/IN Denver/NNP

Can /VBP you/PRP list/VB the/DT nonstop/JJ afternoon/NN flights/NNS

POS tagging methods

- Similar to text classification, we would like to use machine learning methods to do POS tagging.

- Using supervised learning, we need to assemble a text corpus and manually annotate the POS for every word in the corpus (i.e., the Brown corpus) (i.e., the corpus-based methods).
  - We can divide the corpus into training data, development data and test data

- To build a good corpus
  - we must define a task people can do reliably (choose a suitable POS set)
  - we must provide good documentation for the task
    - so annotation can be done consistently
  - we must measure human performance (through dual annotation and inter-annotator agreement)
  - Often requires several iterations of refinement

The simplest POS tagging method

- We tag each word with its most likely part-of-speech (based on the training data)
  - this works quite well: about 90% accuracy when trained and tested on similar texts
  - although many words have multiple parts of speech, one POS typically dominates within a single text type

- How can we take advantage of context to do better?

POS tagger as sequence labeling

- Sequence labeling: given a sequence of observations $x = x_1, x_2, \ldots, x_n$, we need to assign a label/type/class y_i for each observation $x_i \in x$, leading to the sequence label $y = y_1, y_2, \ldots, y_n$ for x ($y_i \in Y$) (Y is the set of possible POS tags)

- For POS tagging, x can be an input sentence where x_i is the i-th word in the sentence, and y_i can be the POS tag of x_i in x (Y is the set of the possible POS tags in our data). E.g.,

x =	Does	this	flight	serve	dinner
y =	VBZ	DT	NN	VB	NN

Sequence labeling

- As in text classification, we also want to estimate the distribution from the training data:

$$P(y|x) = P(y_1, y_2, \ldots, y_n | x_1, x_2, \ldots, x_n)$$

- So, we can also obtain the predicted label sequence for x by:

$$y^* = argmax_y P(y|x) = argmax_y P(y_1, y_2, \ldots, y_n | x_1, x_2, \ldots, x_n)$$

Hidden Markov Model (HMM)

- Using Bayes's Rule

$$argmax_y P(y|x) = argmax_y \frac{P(x|y)P(y)}{P(x)}$$
$$= argmax_y P(x|y)P(y)$$
$$= argmax_c P(x_1, x_2, \ldots, x_n | y_1, y_2, \ldots, y_n) P(y_1, y_2, \ldots, y_n)$$

Prior probability of label sequence

- First-order Markov assumption: the probability of the label for the current step only depends on the label from the previous step, so:

$$P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} P(y_t | y_{<t}) = \prod_{t=1}^{n} P(y_t | y_{t-1})$$

- Independency assumption: the probability of the current word is only dependent on its label:

$$P(x_1, x_2, \ldots, x_n | y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} P(x_t | x_{<t}, y) = \prod_{t=1}^{n} P(x_t | y_t)$$

- So, in HMM, we need to obtain two types of probabilities:
  - The transition probabilities: $P(y_t | y_{t-1})$
  - The emission probabilities: $P(x_t | y_t)$

Parameter Estimation

- Using Maximum Likelihood Estimators as in Naïve Bayes (i.e., just counting):

How many times y_{t-1} and y_t appear together in the training data?

$$P(y_t|y_{t-1}) = \frac{c(y_{t-1}, y_t)}{c(y_{t-1})}$$

How many times y_{t-1} appears in the training data?

$$P(x_t|y_t) = \frac{c(x_t, y_t)}{c(y_t)}$$

How many times x_t appears with y_t in the training data?

- With smoothing:

$$P(y_t|y_{t-1}) = \frac{\alpha + c(y_{t-1}, y_t)}{|Y|\alpha + c(y_{t-1})}$$

$$P(x_t|y_t) = \frac{\alpha + c(x_t, y_t)}{|V|\alpha + c(y_t)}$$

Y is the set of possible POS tags, V is the vocabulary (set of possible words)

How many transition and emission probabilities we have?

Transition probabilities

	NNP	MD	VB	JJ	NN	RB	DT
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

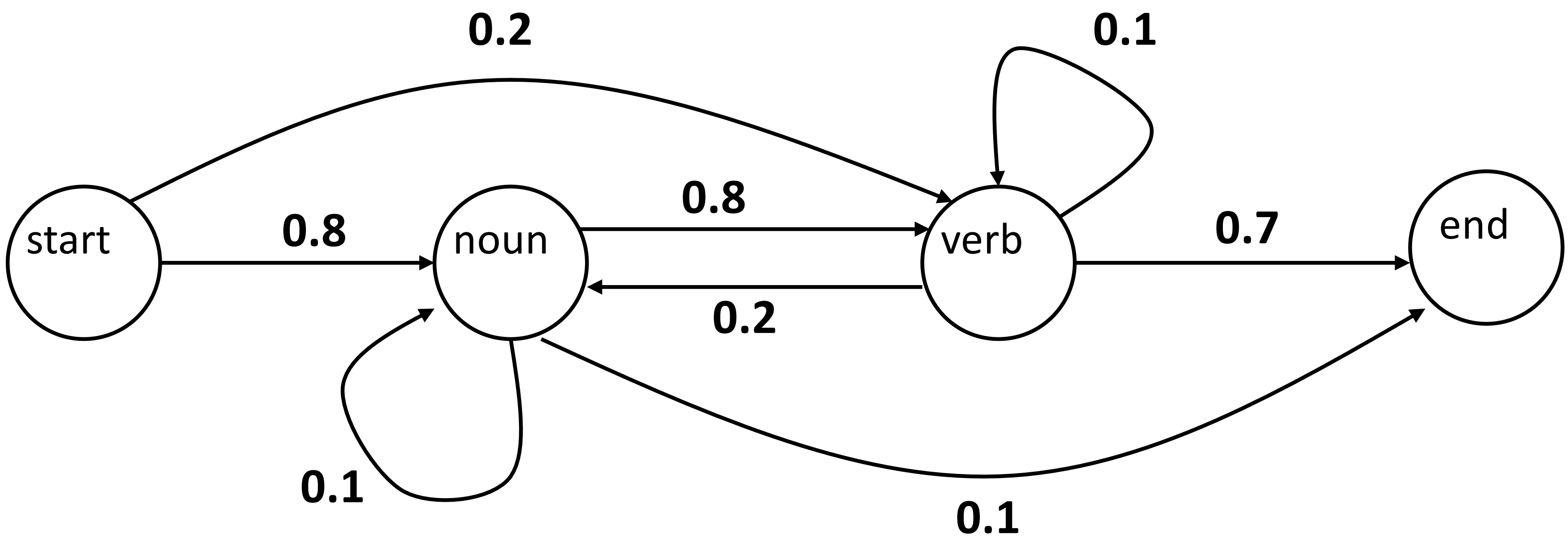
Figure 10.5 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

Emission probabilities

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 10.6 Observation likelihoods *B* computed from the WSJ corpus without smoothing.

Hidden State Network



Decoding

- Given the transition and emission probabilities $P(y_t|y_{t-1})$ and $P(x_t|y_t)$, we need to find the best label sequence $y^* = y_1^*, y_2^*, \dots, y_n^*$ for the input sentence $x = x_1, x_2, \dots, x_n$ via:

$$y^* = argmax_y P(y|x) = argmax_y \frac{P(x,y)}{P(x)} = argmax_y P(x,y)$$
$$= argmax_y P(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n)$$

- This requires the enumeration over all the possible label sequences (paths) y which are exponentially large
  - E.g., using Penn Treebank with 45 tags
    - A sentence of length 5 would have 45^5 = 184,528,15 possible sequences
    - A sentence of length 20 would have 45^{20} = 1.16e33 possible sequences

Greedy Decoder

- simplest decoder (tagger) assign tags deterministically from left to right
- selects y_t^* to maximize $P(x_t|y_t) \, * \, P(y_t|y_{t-1})$
- does not take advantage of right context
- can we do better?

Viterbi algorithm

- Basic idea: if an optimal path through a sequence uses label L at time t, then it must have used an optimal path to get to label L at time t

- We can thus discard all non-optimal paths up to label L at time t

- Let $v_t(s)$ be the probability that the HMM is in state (label) s after seeing the first t observations (words) and passing through the most probable state sequence $y_1, y_2, \ldots, y_{t-1}$:
  $$v_t(s) = max_{y_1,y_2,\ldots,y_{t-1}} P(x_1, x_2, \ldots, x_t, y_1, y_2, \ldots, y_{t-1}, y_t = s)$$

- Introducing the $start$ and end states to represent the beginning and the end of the sentences ($y_0 = start, y_{n+1} = end$), the probability for the optimal label sequence would be:

$$v_{n+1}(end) = max_{y_1,y_2,\ldots,y_n} P(x_1, x_2, \ldots, x_n, y_0 = start, y_1, y_2, \ldots, y_n, y_{n+1} = end)$$

Viterbi algorithm

- $v_t(s) = max_{y_1,y_2,\ldots,y_{t-1}} P(x_1, x_2, \ldots, x_t, y_0 = start, y_1, y_2, \ldots, y_{t-1}, y_t = s)$
- Initialization ($t = 0$):

$$v_0(s) = \begin{cases} 1 \ if \ s = start \\ 0 \ otherwise \end{cases}$$

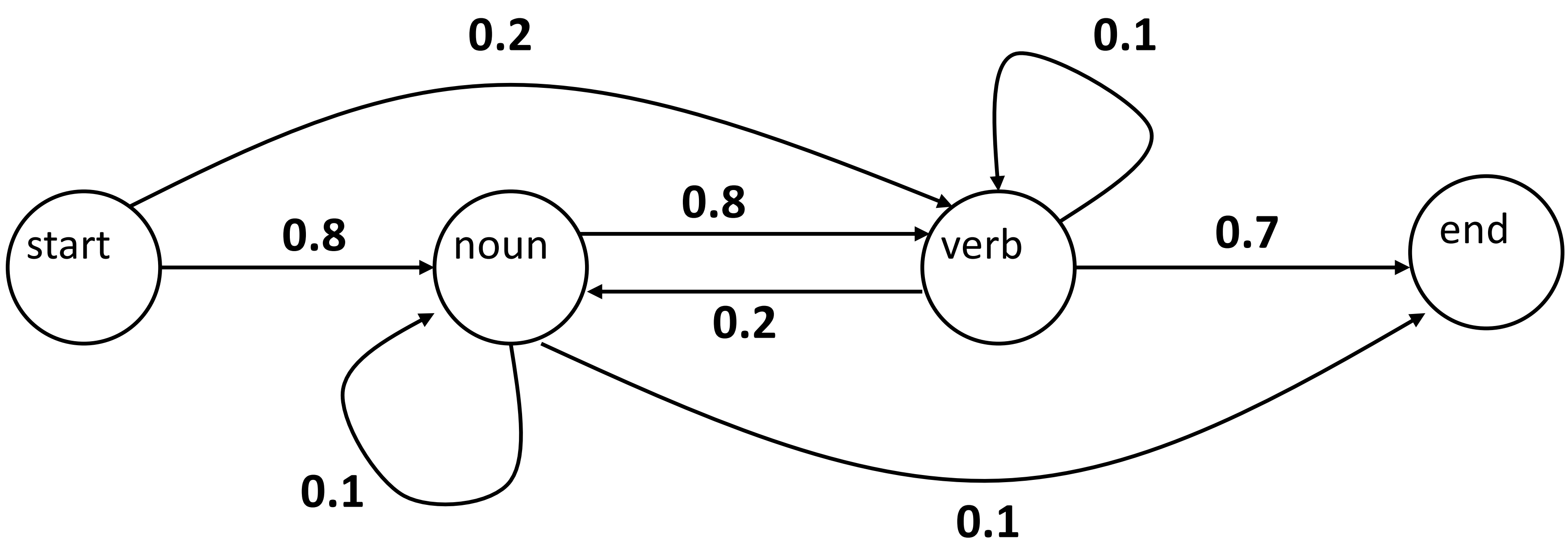
- Recurrence ($t > 0$):

$$v_t(s) = max_{s' \in Y}[v_{t-1}(s')P(s|s')P(x_t|s)]$$
$$backtrack_t(s) = argmax_{s' \in Y}[v_{t-1}(s')P(s|s')P(x_t|s)]$$

- Termination ($t = n + 1$): the optimal probability is $v_{n+1}(end)$, following the backtrack links (starting at $backtrack_{n+1}(end)$) to retrieve the optimal path.

Example

Fish sleep



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 twice 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				

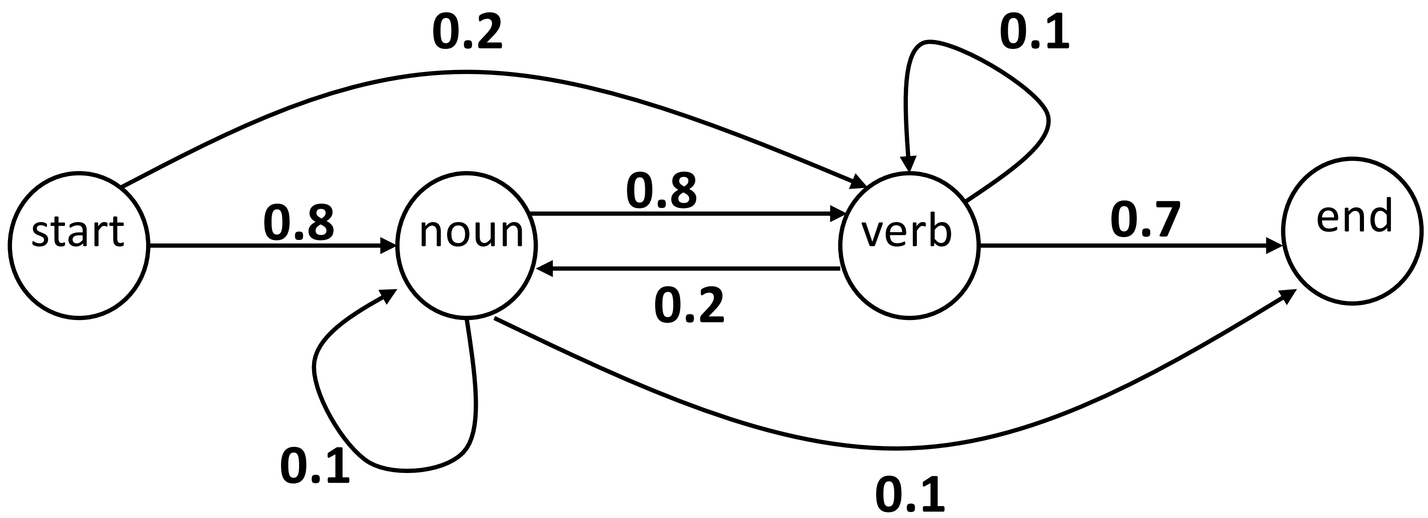
Noun

- P(fish | noun) : 0.8
- P(sleep | noun) : 0.2

Verb

- P(fish | verb) : 0.5
- P(sleep | verb) : 0.5

Init



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

Noun

P(fish | noun) : 0.8

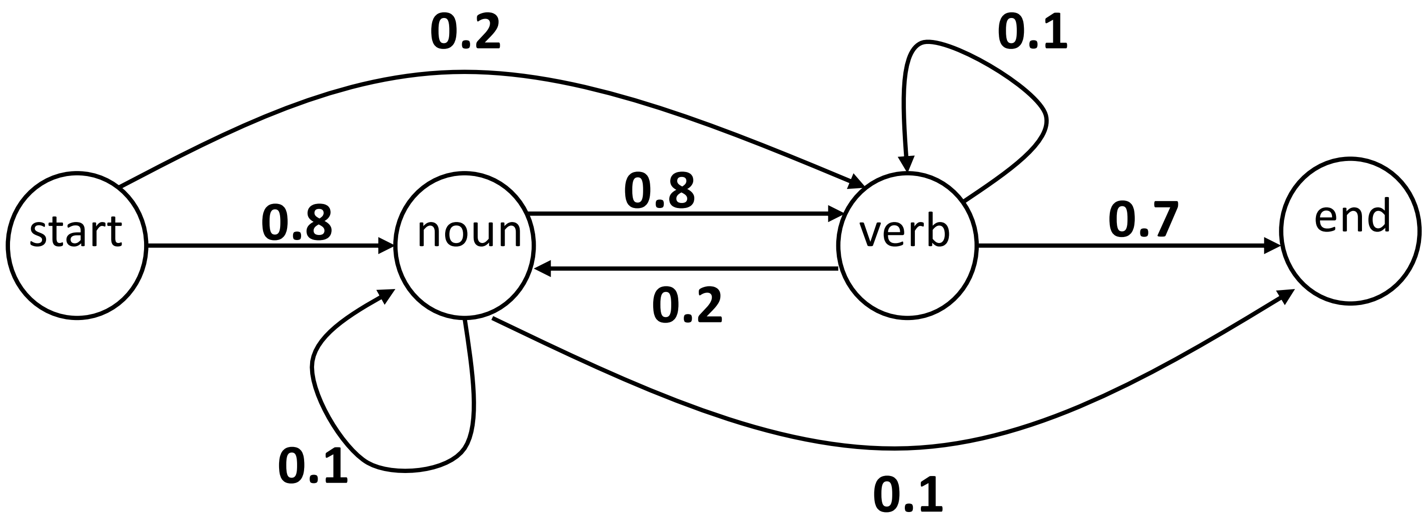
P(sleep | noun) : 0.2

Verb

P(fish | verb) : 0.5

P(sleep | verb) : 0.5

Token 1: fish



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

Noun

P(fish | noun) : 0.8

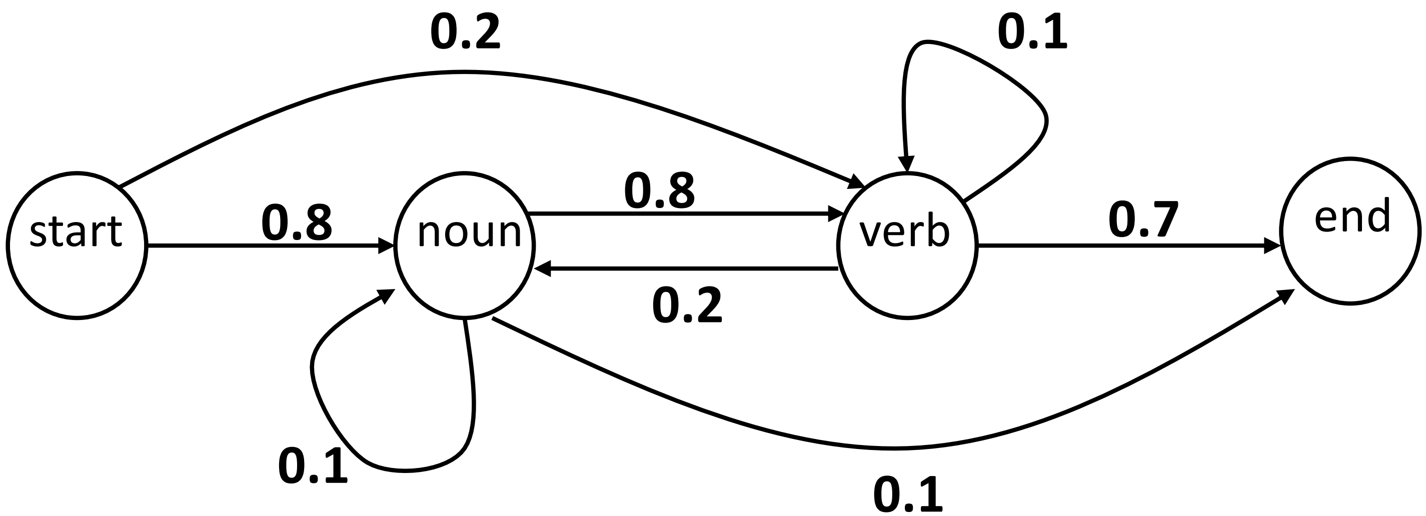
P(sleep | noun) : 0.2

Verb

P(fish | verb) : 0.5

P(sleep | verb) : 0.5

Token 1: fish



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

Noun

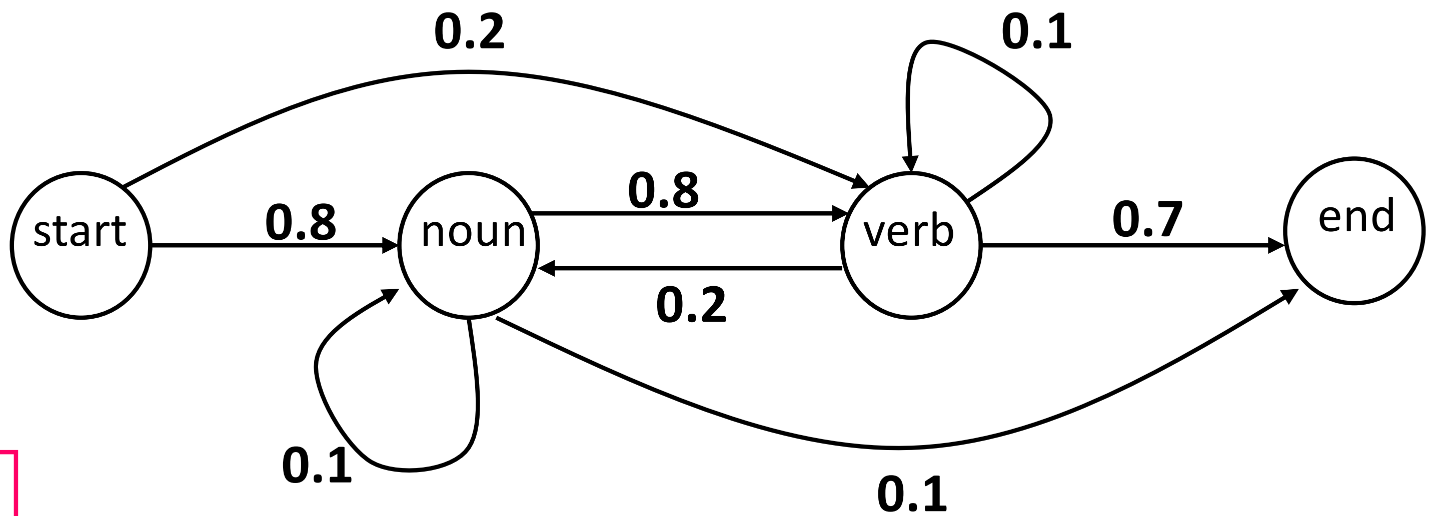
P(fish | noun) : 0.8

P(sleep | noun) : 0.2

Verb

P(fish | verb) : 0.5

P(sleep | verb) : 0.5



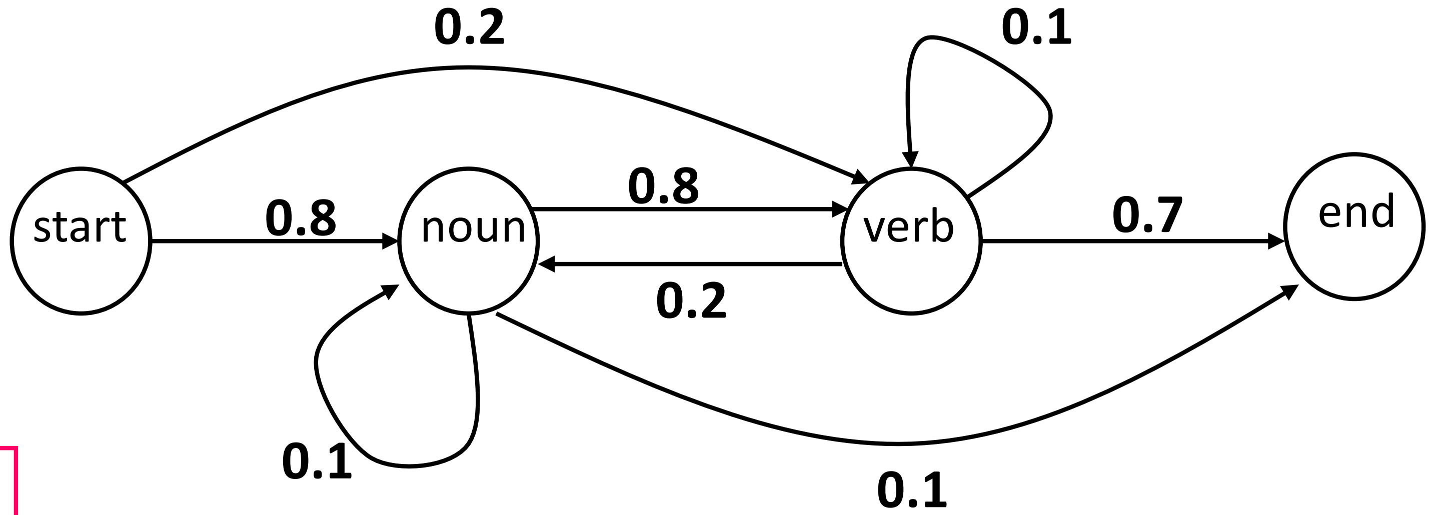
Token 2: sleep

(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	-	

Noun
- P(fish | noun) : 0.8
- P(sleep | noun) : 0.2

Verb
- P(fish | verb) : 0.5
- P(sleep | verb) : 0.5



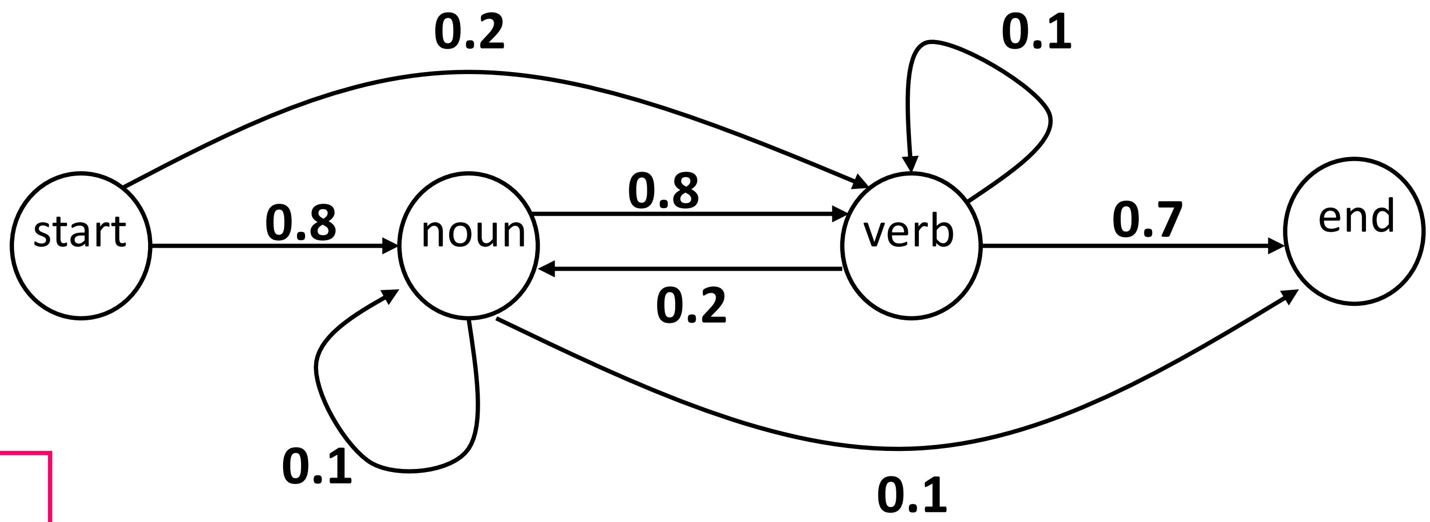
Token 2: sleep

(if 'fish' is verb)

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

Noun
- P(fish | noun) : 0.8
- P(sleep | noun) : 0.2

Verb
- P(fish | verb) : 0.5
- P(sleep | verb) : 0.5



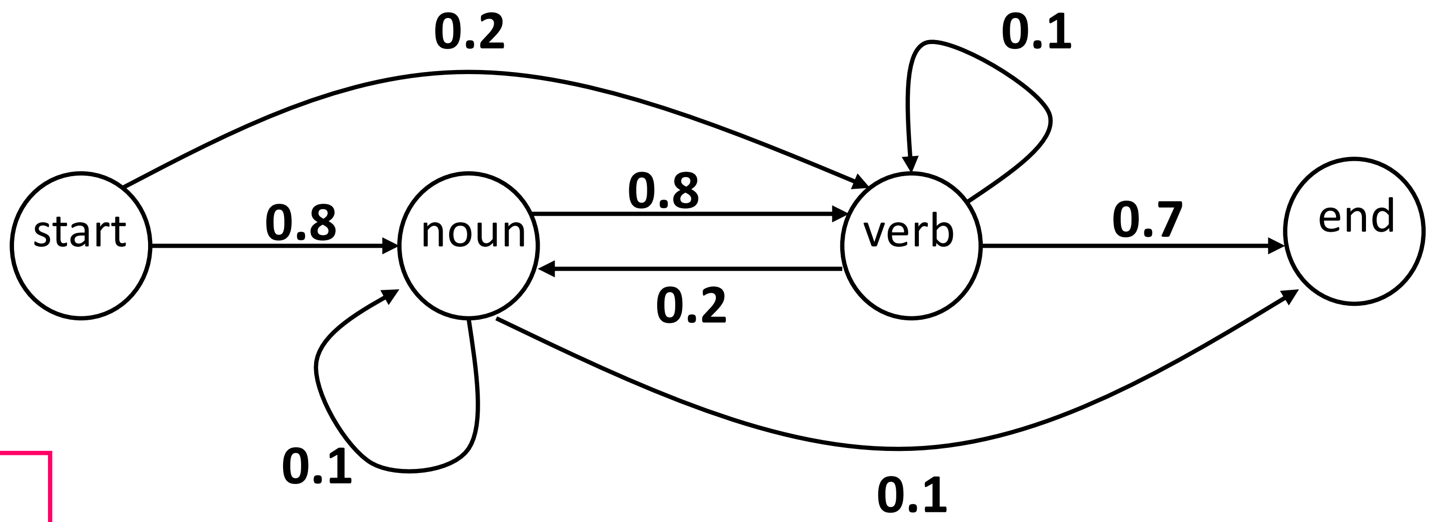
Token 2: sleep

(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	-	

Noun
- P(fish | noun) : 0.8
- P(sleep | noun) : 0.2

Verb
- P(fish | verb) : 0.5
- P(sleep | verb) : 0.5



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	-	

Noun

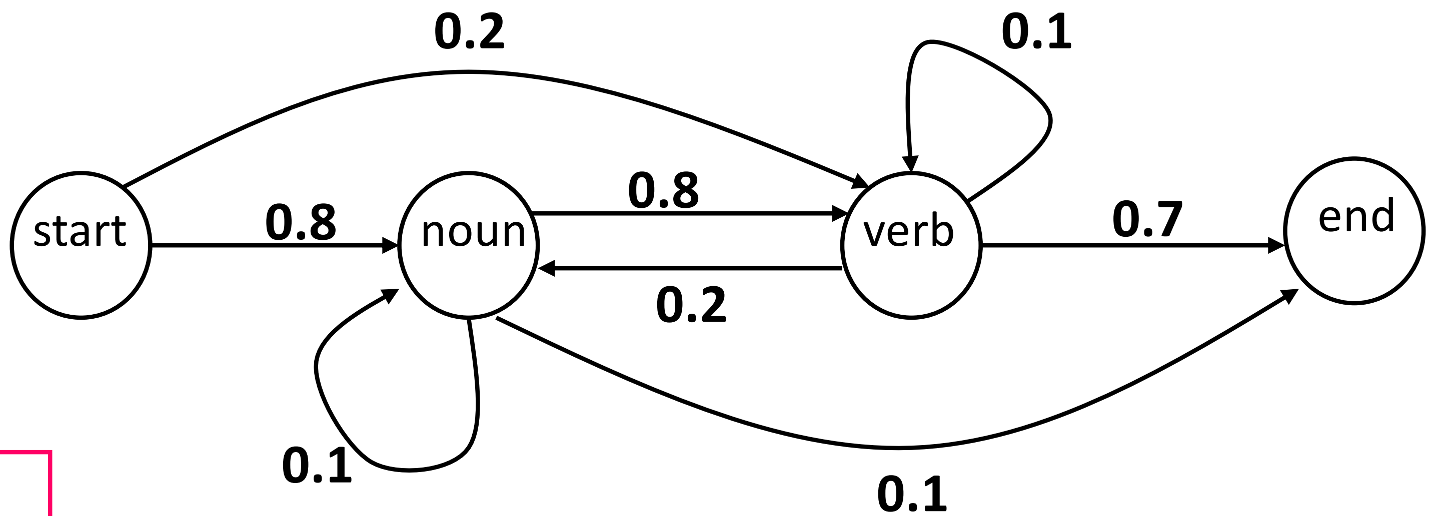
P(fish | noun) : 0.8

P(sleep | noun) : 0.2

Verb

P(fish | verb) : 0.5

P(sleep | verb) : 0.5



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	-	

Noun

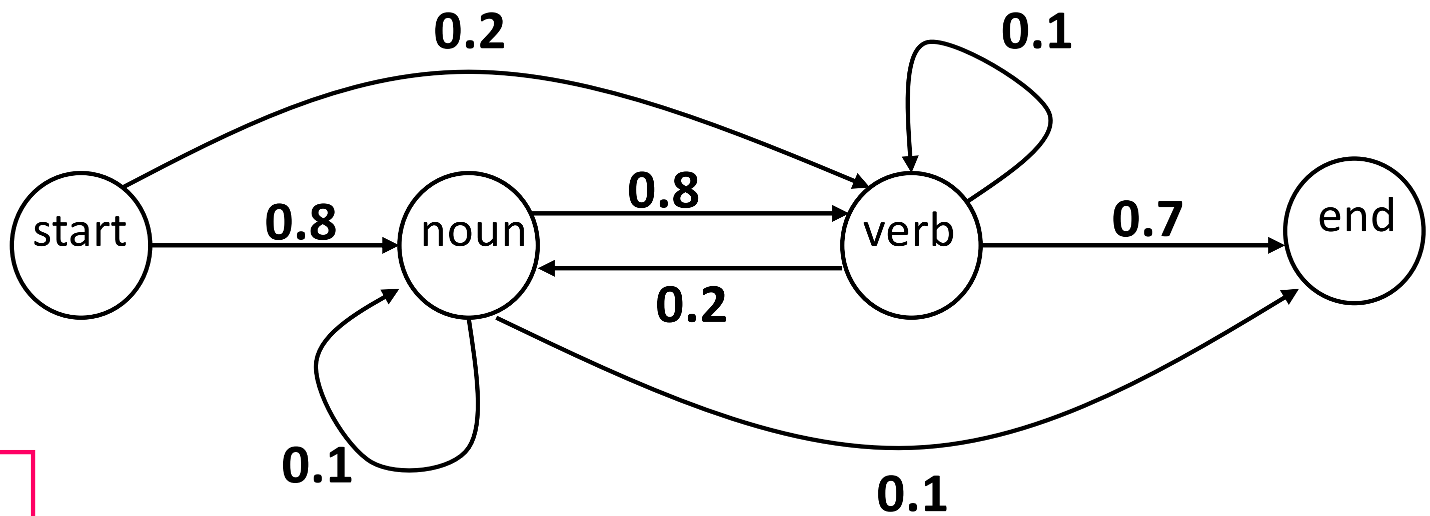
P(fish | noun) : 0.8

P(sleep | noun) : 0.2

Verb

P(fish | verb) : 0.5

P(sleep | verb) : 0.5



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	-	

Noun

P(fish | noun) : 0.8

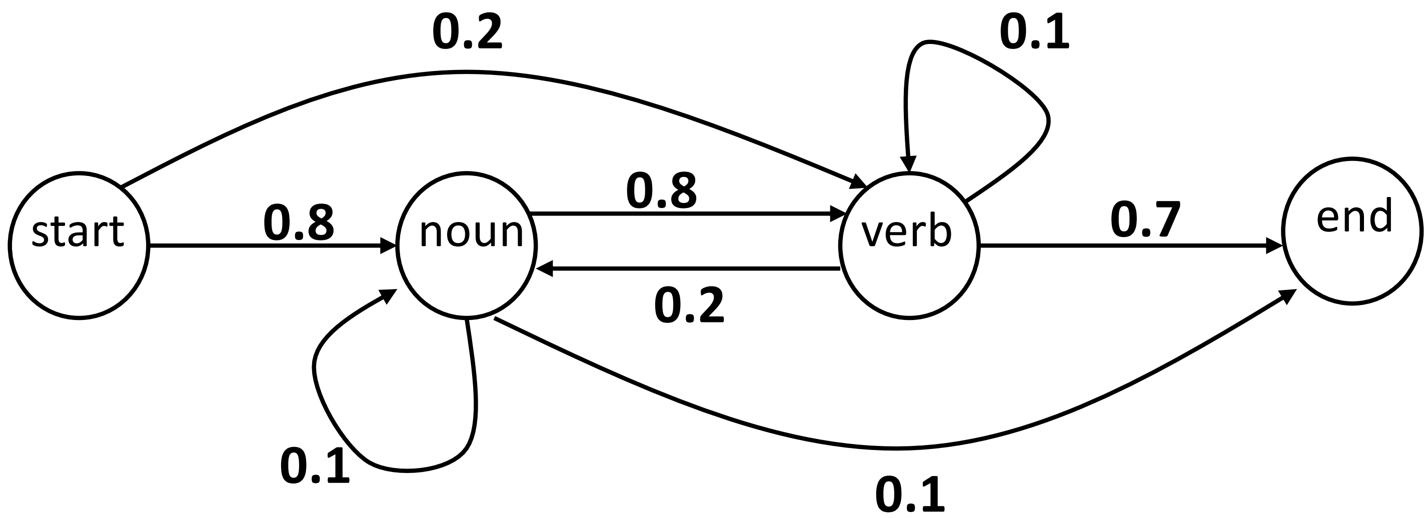
P(sleep | noun) : 0.2

Verb

P(fish | verb) : 0.5

P(sleep | verb) : 0.5

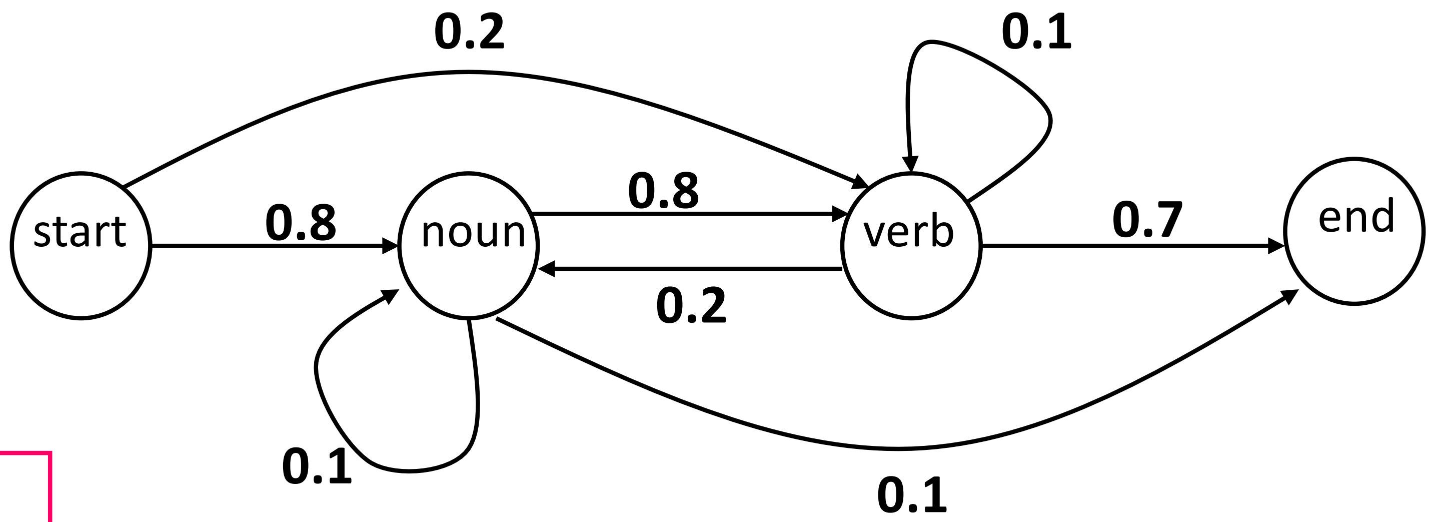
Token 3: end



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

Noun
- P(fish | noun) : 0.8
- P(sleep | noun) : 0.2

Verb
- P(fish | verb) : 0.5
- P(sleep | verb) : 0.5



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~~

Noun

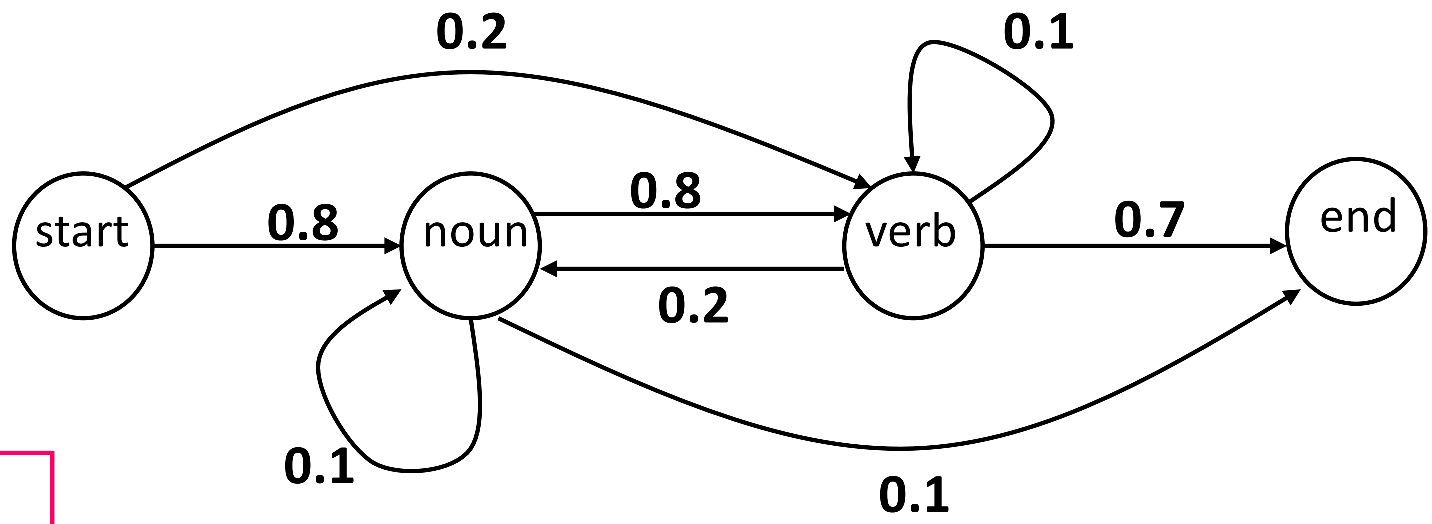
P(fish | noun) : 0.8

P(sleep | noun) : 0.2

Verb

P(fish | verb) : 0.5

P(sleep | verb) : 0.5



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

Complexity for Viterbi

time = $O\ (\ s^2\ n)$

for s states (labels) and n words

(Relatively fast: for 40 states and 20 words, 32,000 steps)

Named Entity Recognition (NER)

- Identify names of entities (i.e., persons, organizations, locations, proteins, etc.) in text.

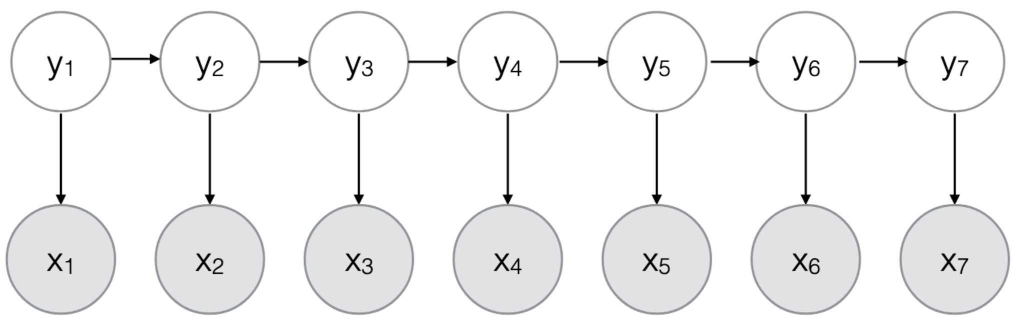
- Can be casted as a sequence labeling problem via the BIO (beginning-inside-other) tagging schema, thus can be solved by HMM

Person				Organization	
Fred	Smith	works	for	Time	inc.
B_PER	I_PER	O	O	B_ORG	I_ORG

HMM for sequence labeling

- simple and fast to train and to use
- effective for POS tagging (one POS ←→ one state)
- can be made effective for name tagging (can capture context) by splitting states
- but further splitting could lead to sparse data problems

$P(y_3 \mid y_2)$



$P(x_3 \mid y_3)$

We want …

- We want to have a more flexible means of capturing our linguistic intuition that certain conditions lead to the increased likelihood of certain outcomes (i.e., feature engineering)
  - that a name on a ‘common first name’ list increases the chance that this is the beginning of a person name
  - that being in a sports story increases the chance of team (organization) names

- Maximum entropy modeling (logistic regression) provides one mathematically well-founded method for combining such features in a probabilistic model.

Maximum Entropy Markov Model (MEMM)

- Starting with the conditional probability distribution:

$$P(y|x) = P(y_1, y_2, \dots, y_n|x) = \prod_{t=1}^{n} P(y_t|y_{<t}, x)$$

- Using the first-order Markov assumption (the probability for the current state only depends on the previous state):

The probability for one step depends on the entire input sentence x

$$P(y_t|y_{<t}, x) \approx P(y_t|y_{t-1}, x)$$

$$P(y|x;\theta) = \prod_{t=1}^{n} P(y_t|y_{<t}, x) \approx \prod_{t=1}^{n} P(y_t|y_{t-1}, x;\theta)$$

- Using logistic regression to model the probabilities $P(y_t|y_{t-1}, x;\theta)$, allowing flexible feature engineering

Maximum Entropy Markov Model (MEMM)

- $P(y_t|y_{t-1}, x)$
- In practice, we even simplify: $P(y_t|y_{t-1}, x) \approx P(y_t|y_{t-1}, x_t)$
- Defining K binary features $f_i(y_{t-1}, x)$ over the the prior label y_{t-1} and the entire input sentence x. For examples:
  - $f_i(y_{t-1}, x) = \begin{cases} 1\ if\ x_i = Smith\ and\ y_{t-1} = \text{B\_PER} \\ 0\ otherwise \end{cases}$
  - $f_i(y_{t-1}, x) = \begin{cases} 1\ if\ x_i\ is\ capitalized \\ 0\ otherwise \end{cases}$
  - $f_i(y_{t-1}, x) = \begin{cases} 1\ if\ x_i\ is\ in\ the\ list\ of\ common\ names\ and\ y_{t-1} = \text{O} \\ 0\ otherwise \end{cases}$
- Then:

$$P(y_t|y_{t-1}, x; \theta) = \frac{\exp(\sum_{i=1}^{K} w_i^{y_t} f_i(y_{t-1}, x))}{Z(y_{t-1}, x)}$$

where Z is the normalizing factor and $w^{y_t} = [w_1^{y_t}, w_2^{y_t}, \dots, w_K^{y_t}]$ is the model parameters specific to y_t.

Maximum Entropy Markov Model (MEMM)

- In order to train the MEMM model (i.e., finding the model parameters), we can also optimize the likelihood function over the training dataset:

$$L(\theta) = -\sum_{(x,y)\in D} \log P(y|x,\theta)$$

- There is no closed-form solution for this optimization problem (as HMM); an iterative solver is required.

- The good thing is the function is convex so easier to solve the those in deep learning. E.g.,
  - Generalized Iterative Scaling (GIS) (https://en.wikipedia.org/wiki/Generalized_iterative_scaling)
  - L-BFGS (https://en.wikipedia.org/wiki/Limited-memory_BFGS)

Feature Engineering

- The main task when using a MaxEnt classifier (e.g., MEMM) is to select an appropriate set of features
  - words in the immediate neighborhood are typical basic features: w_{i-1}, w_i, w_{i+1}
  - patterns constructed for rule-based taggers are likely candidates: w_{i+1} is an initial
  - membership on word lists: w_i is a common first name (from Census)

Greedy decoding for MEMM

- At $i = 0$, select:

$$y_1^* = argmax_s P(y_1 = s | y_0 = start, x) = argmax_s P(y_1 = s | x)$$

- At $i > 0$, select:

$$y_i^* = argmax_s P(y_i = s | y_{i-1} = y_{i-1}^*, x)$$

Note that we need to condition on the predicted label from the previous step y_{i-1}^* here as this is now known in the inference/test time.

Viterbi decoding for MEMM

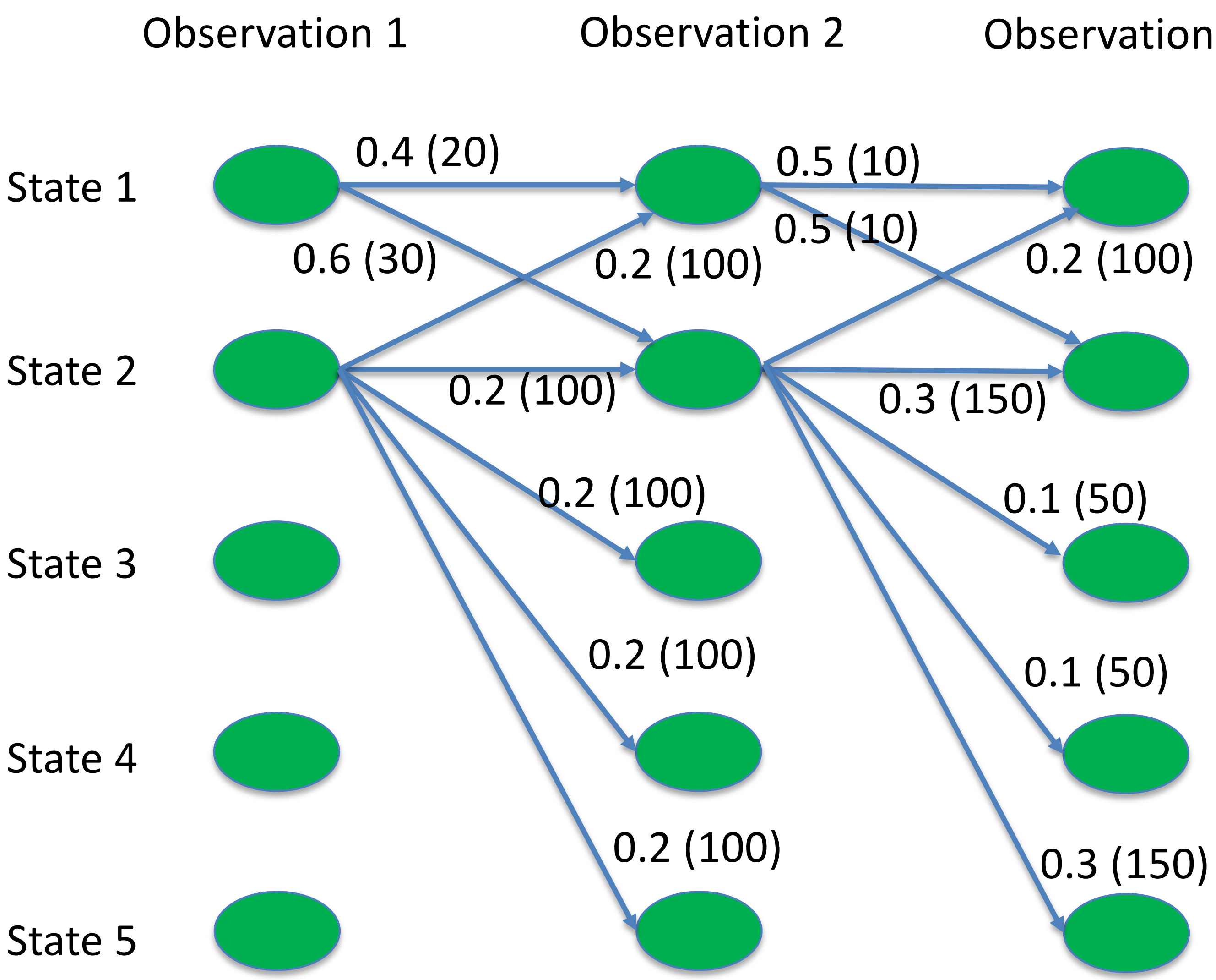
- In HMM, we infer the best label sequence via the joint probability $argmax_y P(x, y)$ using the recurrence:

$$v_t(s) = max_{s' \in Y}[v_{t-1}(s')\ P(y_t = s|y_{t-1} = s')P(x_t|y_t = s)]$$

- In MEMM, we infer the best label sequence via the conditional probability $argmax_y P(y|x)$ using the recurrence:

$$v_t(s) = \ max_{y_1, y_2, \ldots, y_{t-1}} P(y_1, y_2, \ldots, y_{t-1}, y_t = s|x)$$
$$v_t(s) = max_{s' \in Y}[v_{t-1}(s')\ P(y_i = s|y_{i-1} = s', x)]$$
$$p^* = \ max_{s \in Y}\, v_n(s)$$

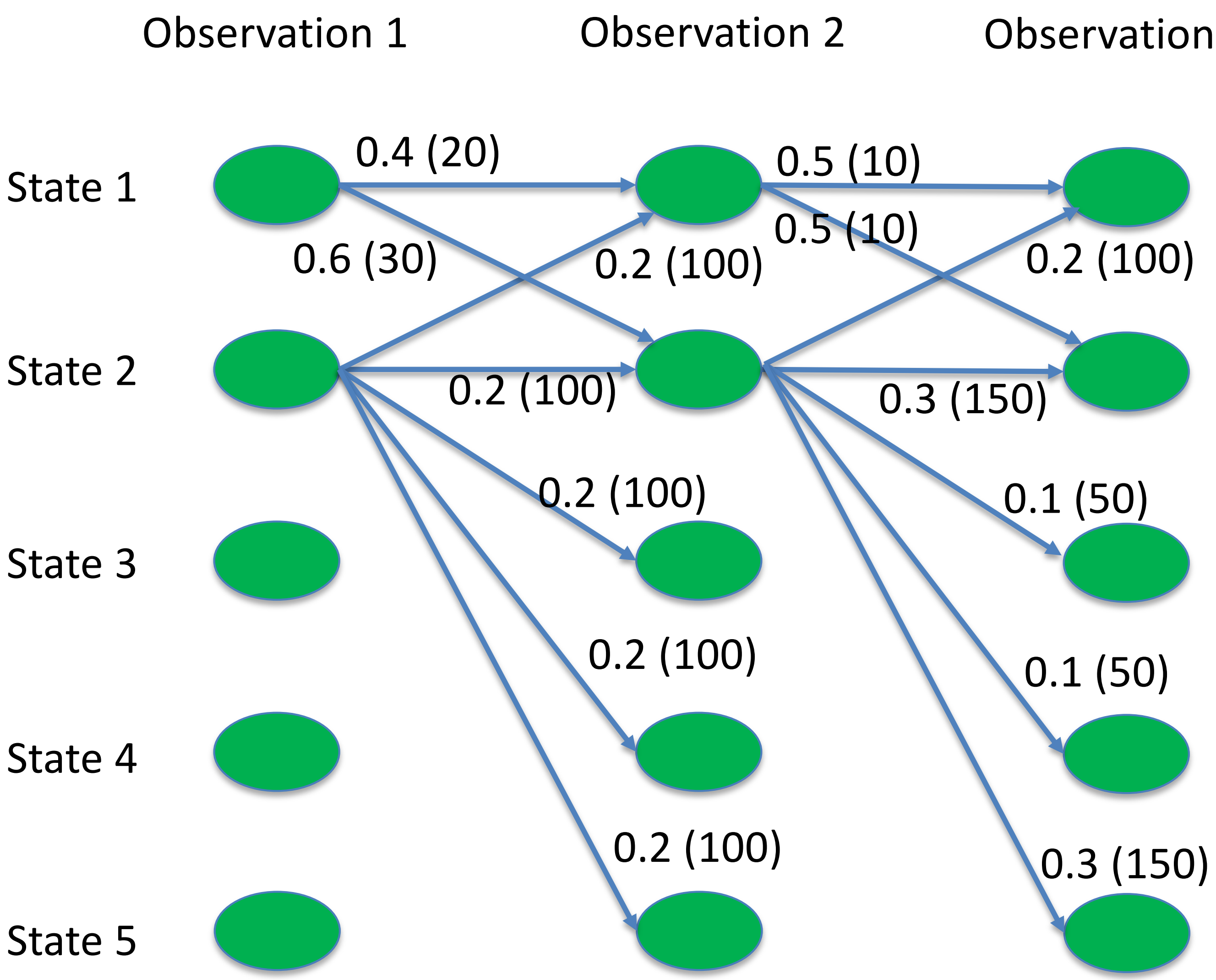
The label bias problem in MEMM



- The scores in the bracket represent the ability to go from one state to another state given the observation, i.e., $\exp(\sum_{i=1}^{K} w_i^{y_t} f_i(y_{t-1}, x))$
- Based on these scores, the best paths should be: 2 -> 2 -> 2 or 2 -> 2 -> 5
- However, if we normalize at each state to obtain the probabilities, the best paths should be: 1 -> 1 -> 1 or 1 -> 1 -> 2

1 -> 1 -> 1, 1 -> 1 -> 2: 0.4 * 0.5 = 0.2
2 -> 2 -> 2, 2 -> 2 -> 5: 0.2 * 0.3 = 0.06

The label bias problem in MEMM



- This is because the prediction at each state/word is modeled by a probability, thus necessitating the normalization at each state (local normalization)
  - Impose a preference of states with lower number of transitions over the others.
- So, we want to avoid the normalization at each step and only normalize once over the entire input sequence to obtain the overall probability $P(y|x)$ (global normalization), leading to Conditional Random Fields (CRF)

Conditional Random Fields (CRF)

- Both MEMM and CRF directly model $P(y|x)$.

- For MEMM:

$$P(y|x;\theta) = \prod_{t=1}^{n} P(y_t|y_{t-1}, x; \theta)$$

- For CRF:

$$P(y|x;\theta) = \frac{\exp(\Phi(x,y)^T\theta)}{\sum_{y' \in Y} \exp(\Phi(x,y')^T\theta)}$$

Conditional Random Fields (CRF)

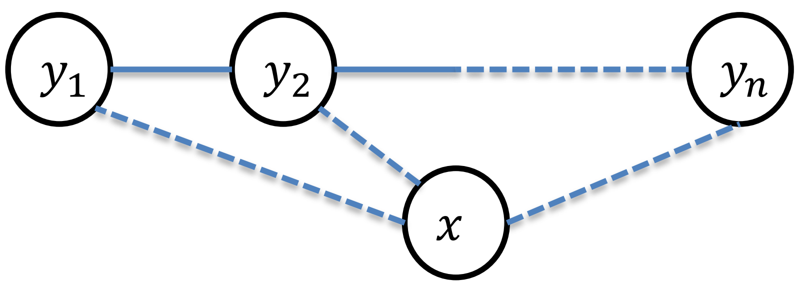
- $P(y|x;\theta) = \frac{\exp(\Phi(x,y)^T\theta)}{\sum_{y' \in Y} \exp(\Phi(x,y')^T\theta)} = \frac{\exp(\Phi(x,y)^T\theta)}{Z(x)}$

where

$$\Phi(x,y) = [\Phi_1(x,y), \dots, \Phi_k(x,y), \dots, \Phi_K(x,y)]$$
$$\Phi_k(x,y) = \sum_{i=1..n} \phi_k(y_{i-1}, y_i, x, i)$$

with $\phi_k(y_{i-1}, y_i, x, i)$ is a function to capture some features of the input sentence x and the transition from state y_{i-1} to state y_i at step i (i.e., only capturing features at the edge and node level and similar to those we use for MEMM).

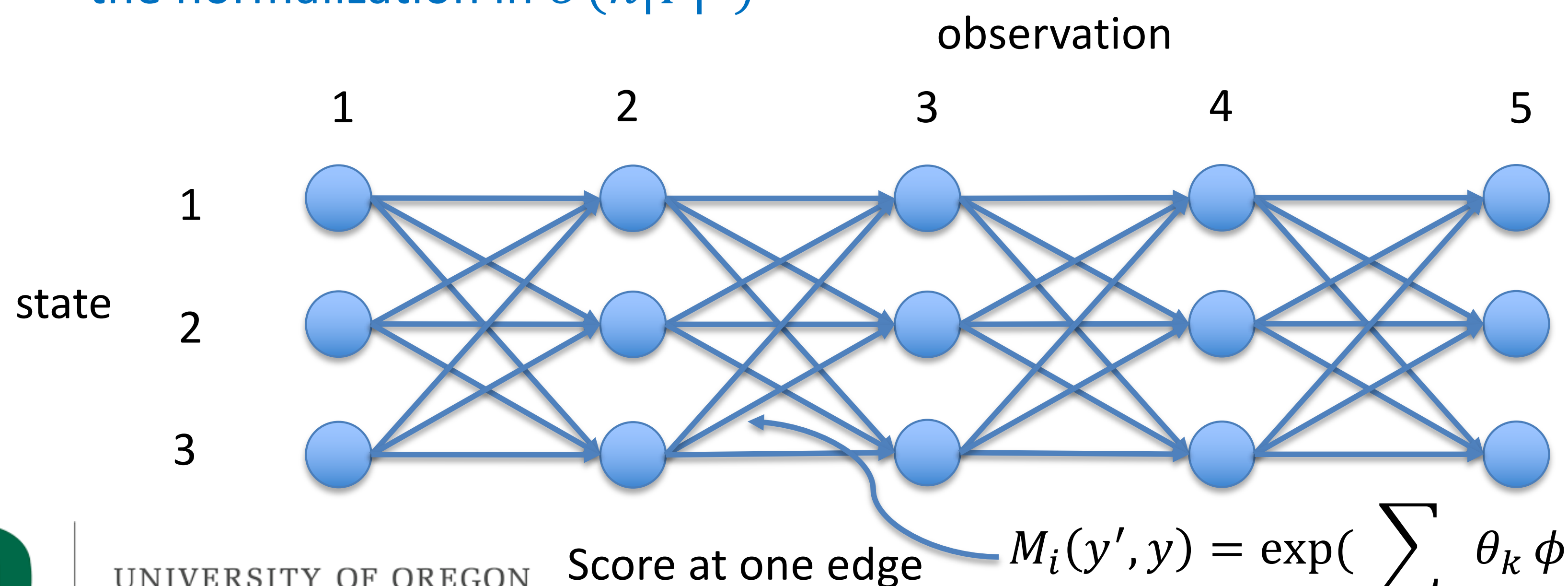
- The element of θ corresponding to $\Phi_k(x,y)$ is θ_k



Conditional Random Fields (CRF)

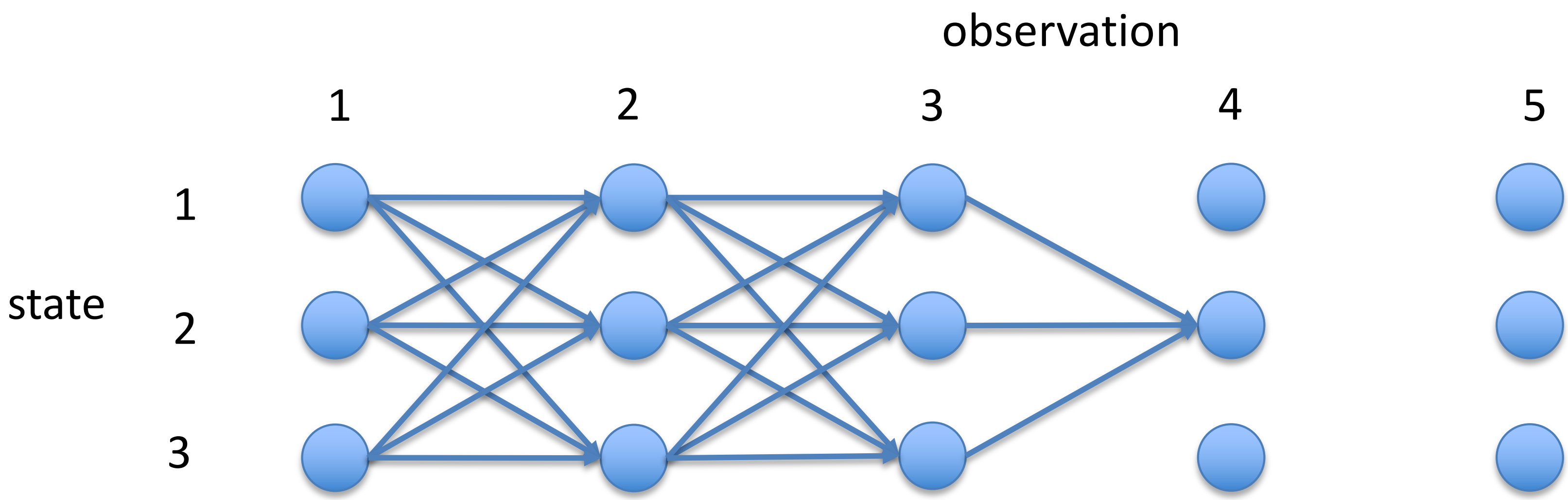
- $P(y|x;\theta) = \frac{\exp(\Phi(x,y)^T\theta)}{\sum_{y' \in Y} \exp(\Phi(x,y')^T\theta)} = \frac{\exp(\Phi(x,y)^T\theta)}{Z(x)}$

- The normalizing factor $Z(x)$ involve summing over an exponential number of terms (all the possible label sequence for the input sentence -- $|Y|^n$)
- Using dynamic programming (i.e., the forward algorithm), we can compute the normalization in $O(n|Y|^2)$



$$M_i(y', y) = \exp\Big(\sum_{k=1..K} \theta_k\, \phi_k(y', y, x, i)\Big)$$

Conditional Random Fields (CRF)



- $\alpha_i(s)$: the total score for the length-i subpaths of the paths whose i-th state is s.

- Initialization:
  $\alpha_1(s) = \exp(\sum_{k=1..K} \theta_k \, \phi_k(start, s, x, 1)$
- Recurrence:
  $\alpha_i(s) = \sum_{s' \in Y} \alpha_{i-1}(s') M_i(s', s)$
- Final normalization score:
  $Z(x) = \sum_{s \in Y} \alpha_n(s)$

CRF Training

- Loss function:

$$L(\theta) = -\log P(y|x;\theta) = -\log \frac{\exp(\Phi(x,y)^T\theta)}{\sum_{y' \in Y} \exp(\Phi(x,y')^T\theta)} = -\Phi(x,y)^T\theta + \log Z(x)$$

- In most of the optimization technique for $L(\theta)$, we will need to compute its gradient:

$$\frac{\partial L(\theta)}{\partial \theta_k} = -\phi_k(x,y) + \sum_{y' \in Y} \frac{\exp(\Phi(x,y')^T\theta)\phi_k(x,y')}{Z(x)} = -\phi_k(x,y) + \boxed{\sum_{y' \in Y} P(y'|x)\phi_k(x,y')}$$

- $\sum_{y' \in Y} P(y'|x)\phi_k(x,y') =$
  $\sum_{i=1..n} \sum_{s' \in Y, s \in Y} \phi_k(s',s,x,i) \sum_{y': y'_{i-1}=s', y'_i=s} P(y'|x)$
- Using this factorization, we can compute this quantity in $O(n|Y|^2)$ using the forward-backward algorithm

For details, see: Collins, “The Forward-Backward Algorithm”

Viterbi decoding for CRF

- $v_t(s) = \ max_{y_1,y_2,\dots,y_{t-1}} P(y_1, y_2, \dots, y_{t-1}, y_t = s|x)$

- Initialization:

  $v_1(s) = \sum_{k=1..K} \exp(\theta_k \phi_k(start, s, x, 1))$

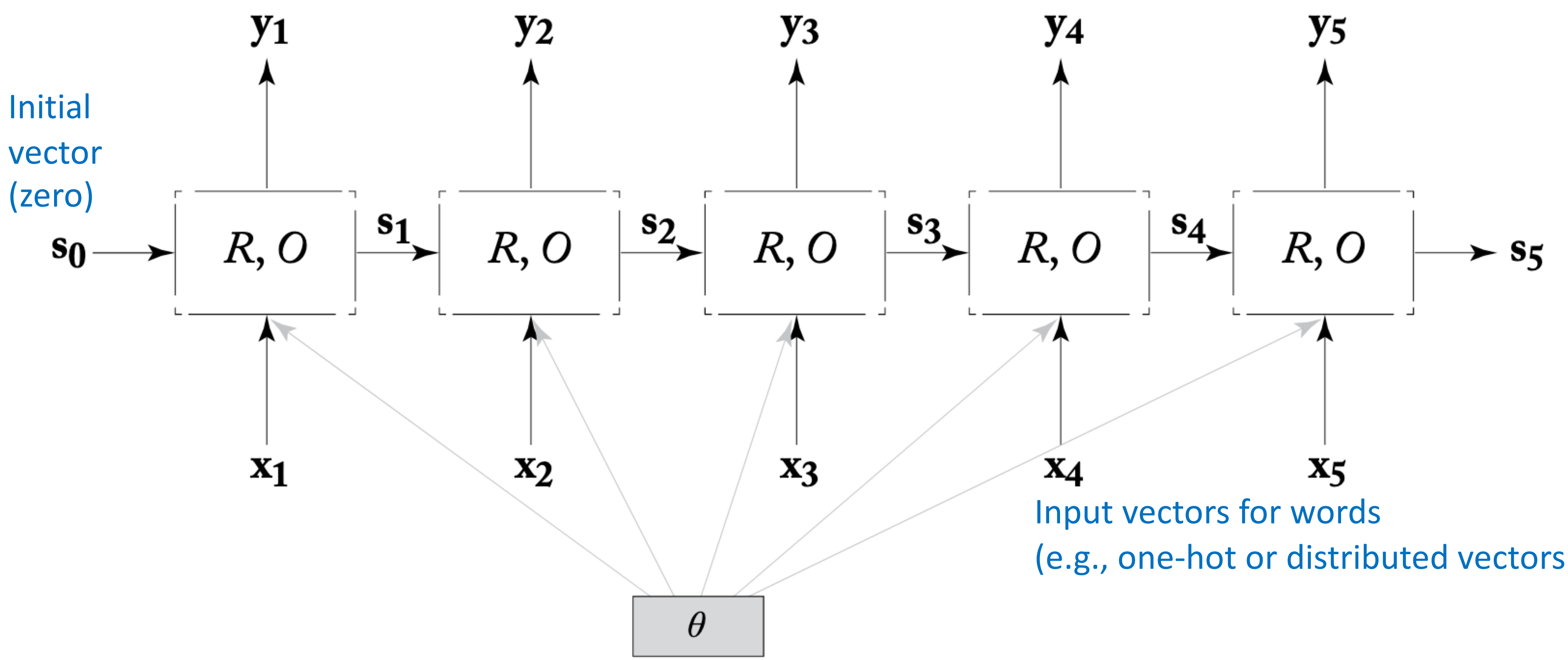
- Recurrence:

  $v_i(s) = max_{s' \in Y}[\alpha_{i-1}(s') M_i(s', s)]$

- Best score:

  $p^* = \ max_{s \in Y} v_n(s)$

Recurrent Neural Networks (RNN)



- R: recurrence function
- O: output function
- s_i, y_i: hidden vector and output vector at step i.
- θ: model parameters (to be learned during training)

Goldberg, 2017

Recurrent Neural Networks (RNN)

- At each step, the R function takes two inputs (i.e., the hidden vector from the previous step s_{t-1} and the input vector from the current step x_t) to compute the hidden vector for the current step s_t :

$$s_t = R(s_{t-1}, x_t)$$

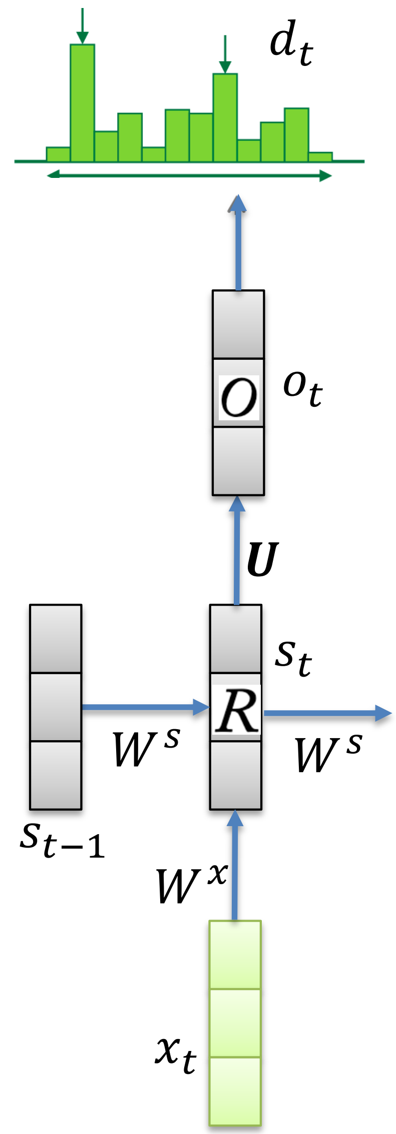
- The hidden vector s_t can be used as the feature vector to make a prediction about the label for x_t (i.e., POS or NER). Essentially, we use the O function to transform s_t into a score vector o_t whose dimensions quantify the likelihood that x_t has the corresponding labels (i.e., $|o_t| = |Y|$):

$$o_t = O(s_t W^o + b^o)$$

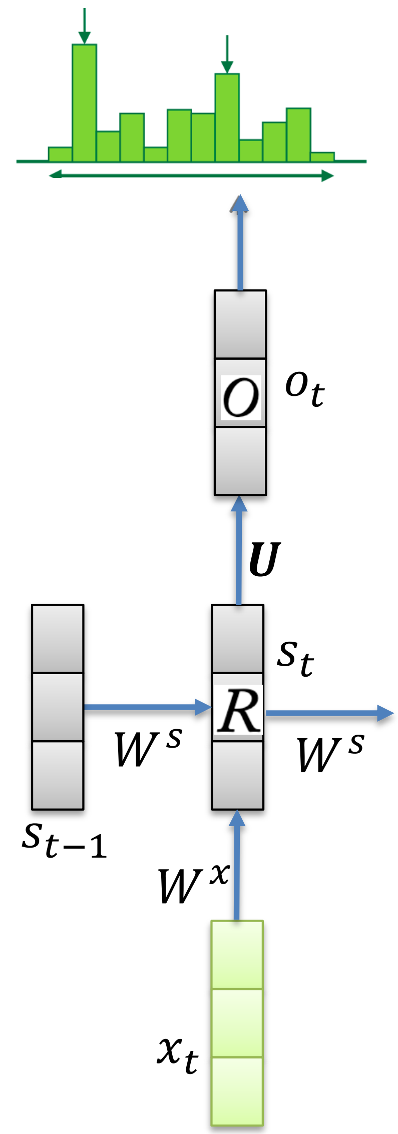
- o_t can be transformed into a probability distribution via the softmax function: $d_t = softmax(o_t)$

- In the simplest version (i.e., vanilla RNN), O can be just the identity function (i.e., $O(x) = x$), while R can be a simple linear transformation followed by a non-linear function:

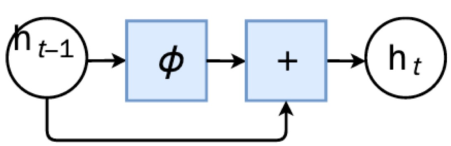
$$s_t = \sigma(s_{t-1} W^s + x_t W^x + b^s)$$



Recurrent Neural Networks (RNN)

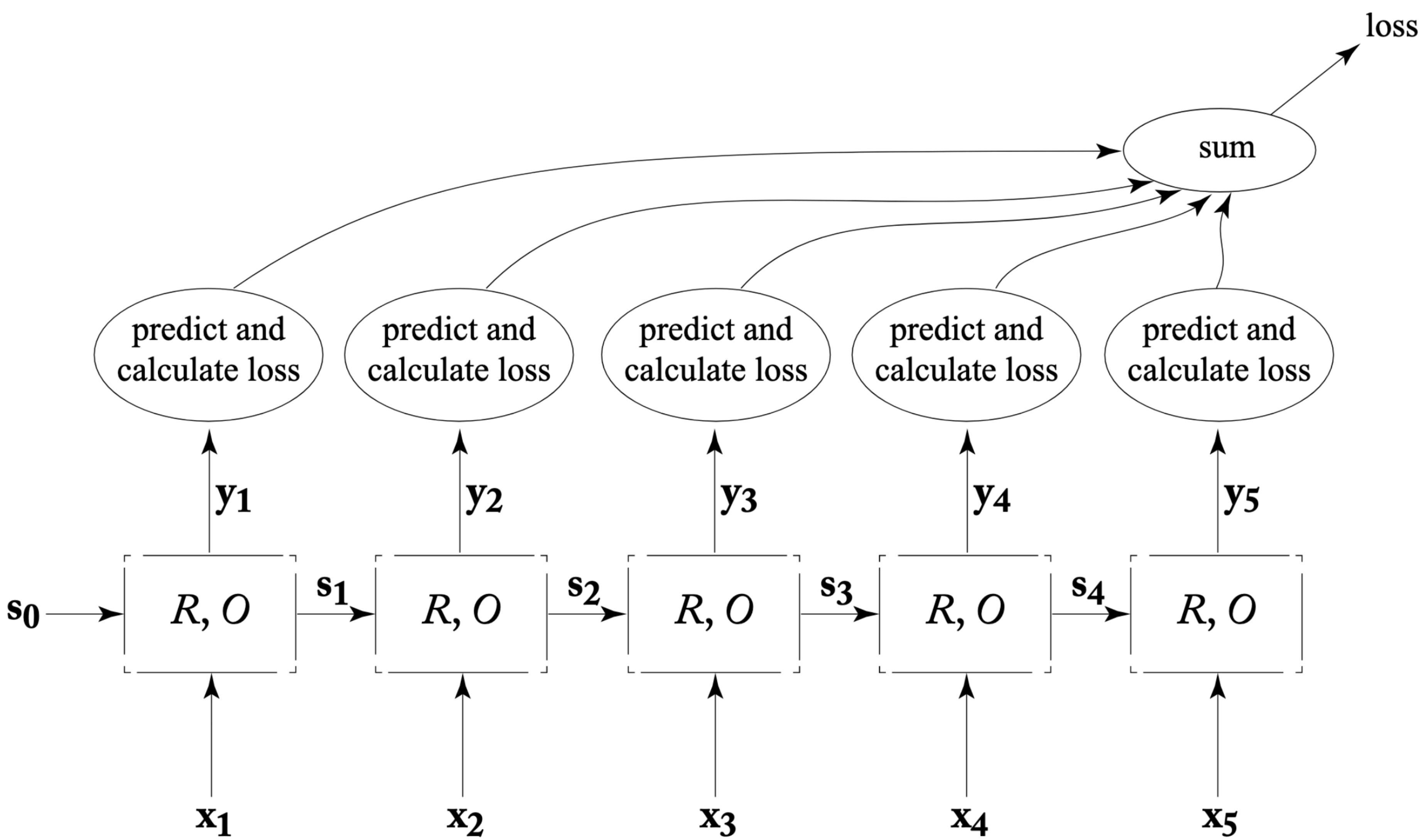


- The model parameters: $\theta = \{W^s, W^x, b^s, W^o, b^o\}$
- The recurrence nature (i.e., using the hidden vector from the previous step for the current computation) allows each hidden vector s_t to capture information about all the words before t: $s_t = f(s_0, s_1, \dots, s_{t-1})$
- The use of the same parameters W^s, W^x, b^s in the recurrence function R causes the *gradient vanishing* problem (i..e, gradient becomes small in long sentences so the models cannot learn)
- In practice, the LSTM cell is often used for R to mitigate this problem.



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

Training RNN



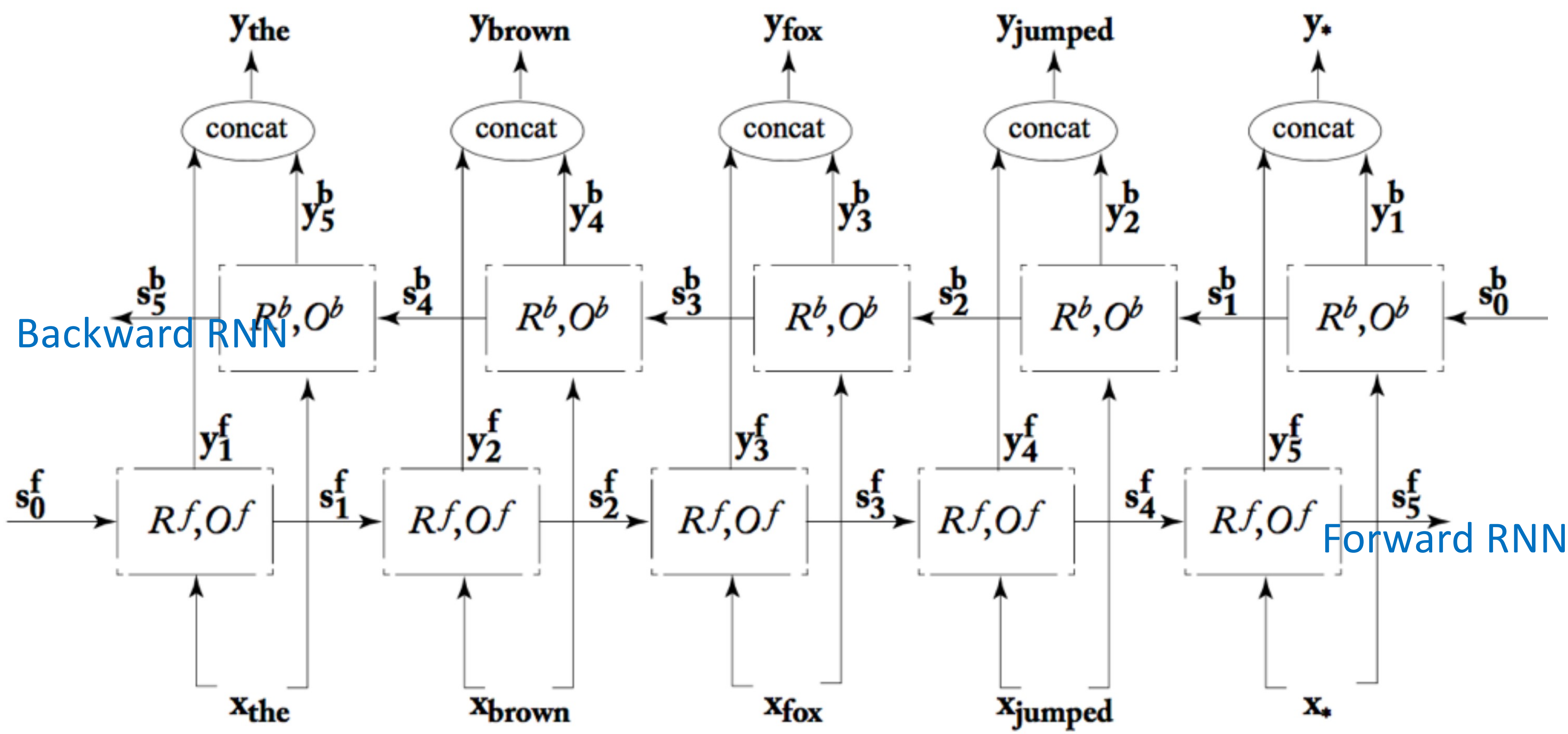
Bidirectional RNN

A city or a football team?

Liverpool suffered an upset first home league defeat of the season, beaten 1-0 by a Guy Whittingham goal for Sheffield Wednesday.

- The information on the left is not enough to predict the label for the current word.

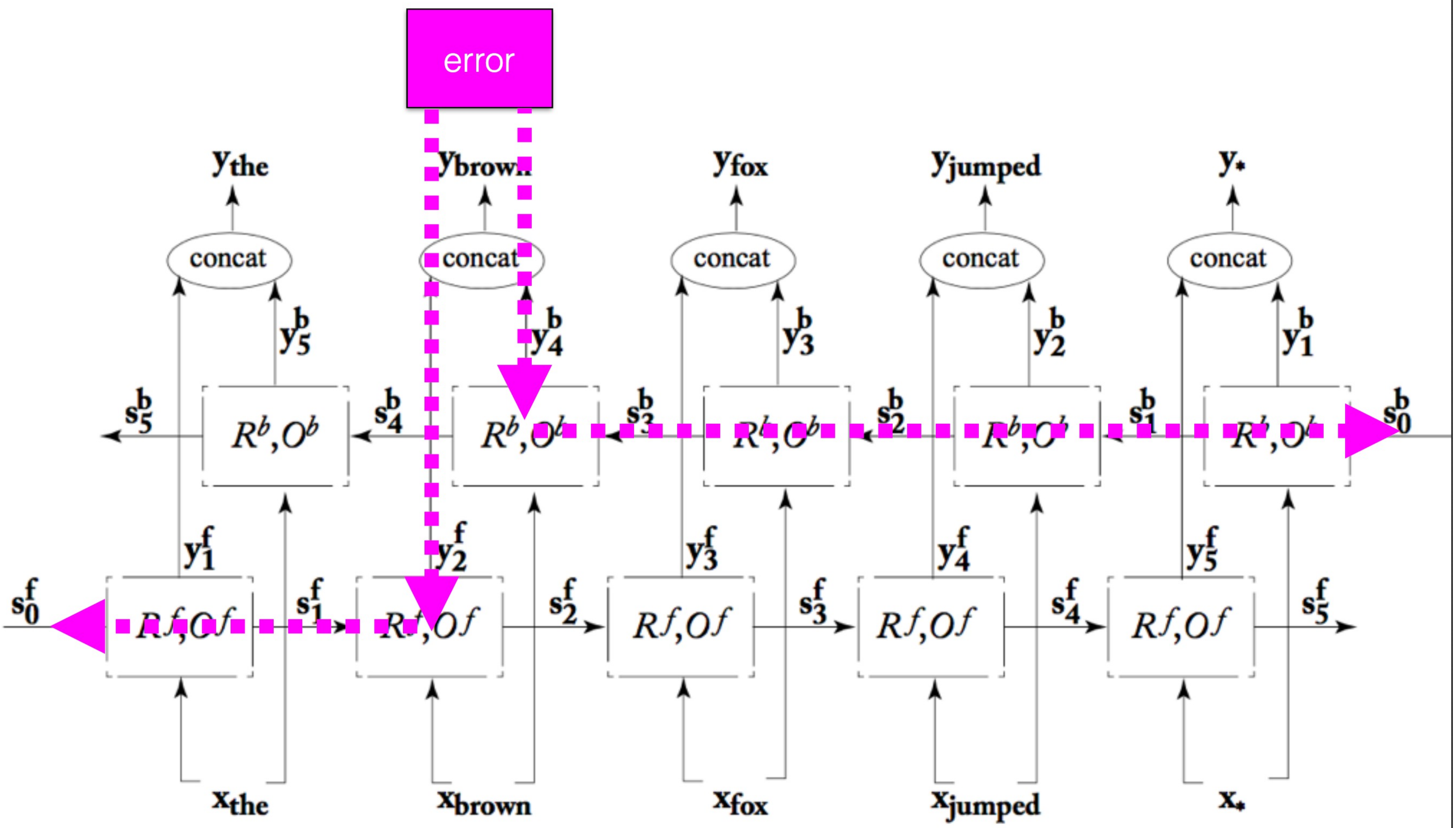
Bidirectional RNN



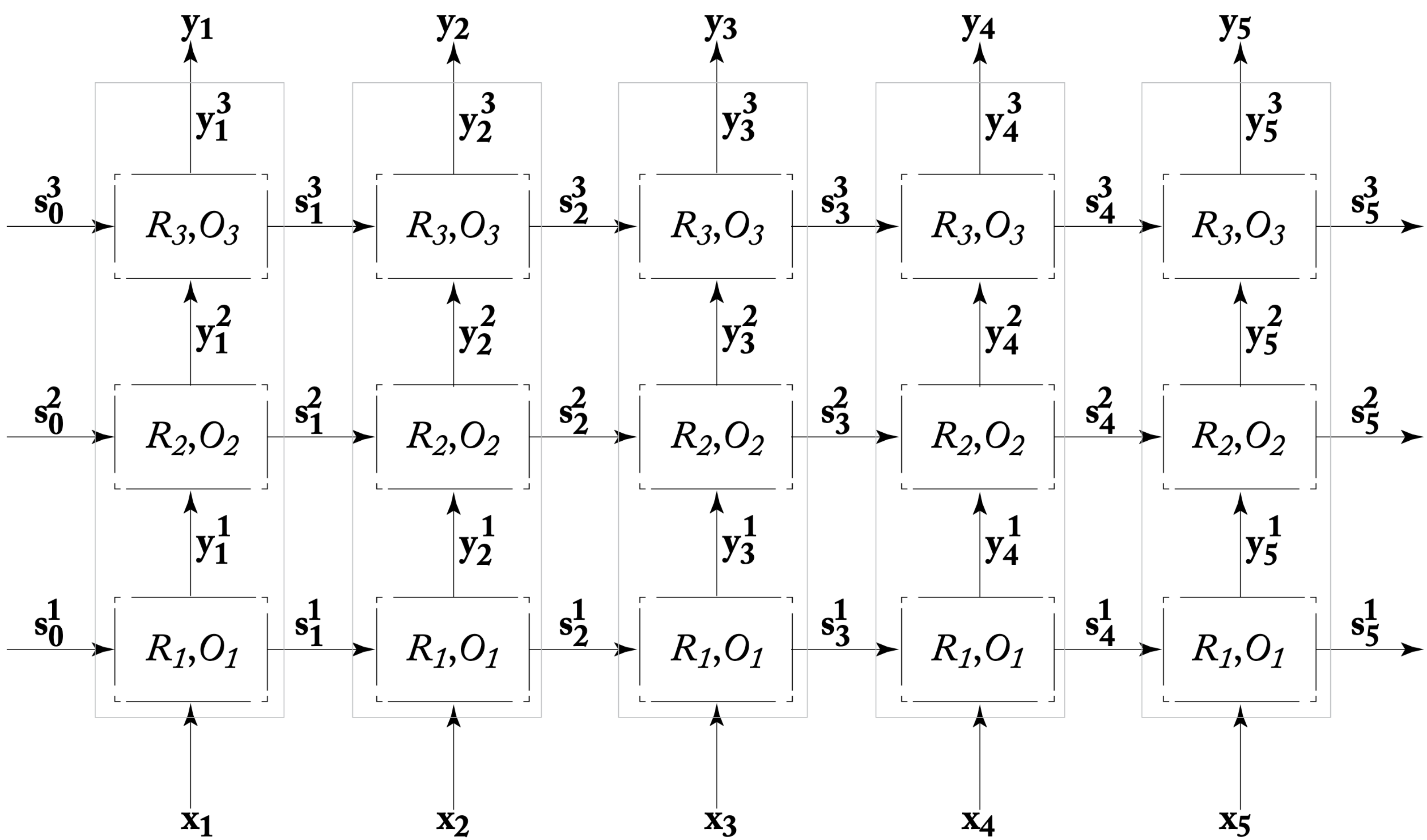
- $s_i^f = \sigma(s_{i-1}^f W_s^f + x_i W_x^f + b^f)$
- $s_i^b = \sigma(s_{i-1}^b W_s^b + x_i W_x^b + b^b)$
- $y_i = softmax([s_i^f, s_i^b] W^o + b^o), \theta = [W_s^f, W_x^f, b^f, W_s^b, W_x^b, b^b, W^o, b^o]$

So, one hidden vector has access to the context information across the whole sentence

Bidirectional RNN



We can also go deeper (stacked RNN)



Incorporating CRF

- RNN makes prediction for words independently (the features/representations share the parameters, but the output predictions are independent)

- There are some dependencies between the output labels that we want to exploit (i.e., I_PER can only be preceded by B_PER), so the later predictions can influence the prior predictions (e.g., fixing prior's error)

- CRF can achieve this via the global normalization of the label sequence probabilities

- Idea: Incorporate CRF as the final in the RNN models for sequence labeling

Incorporating CRF

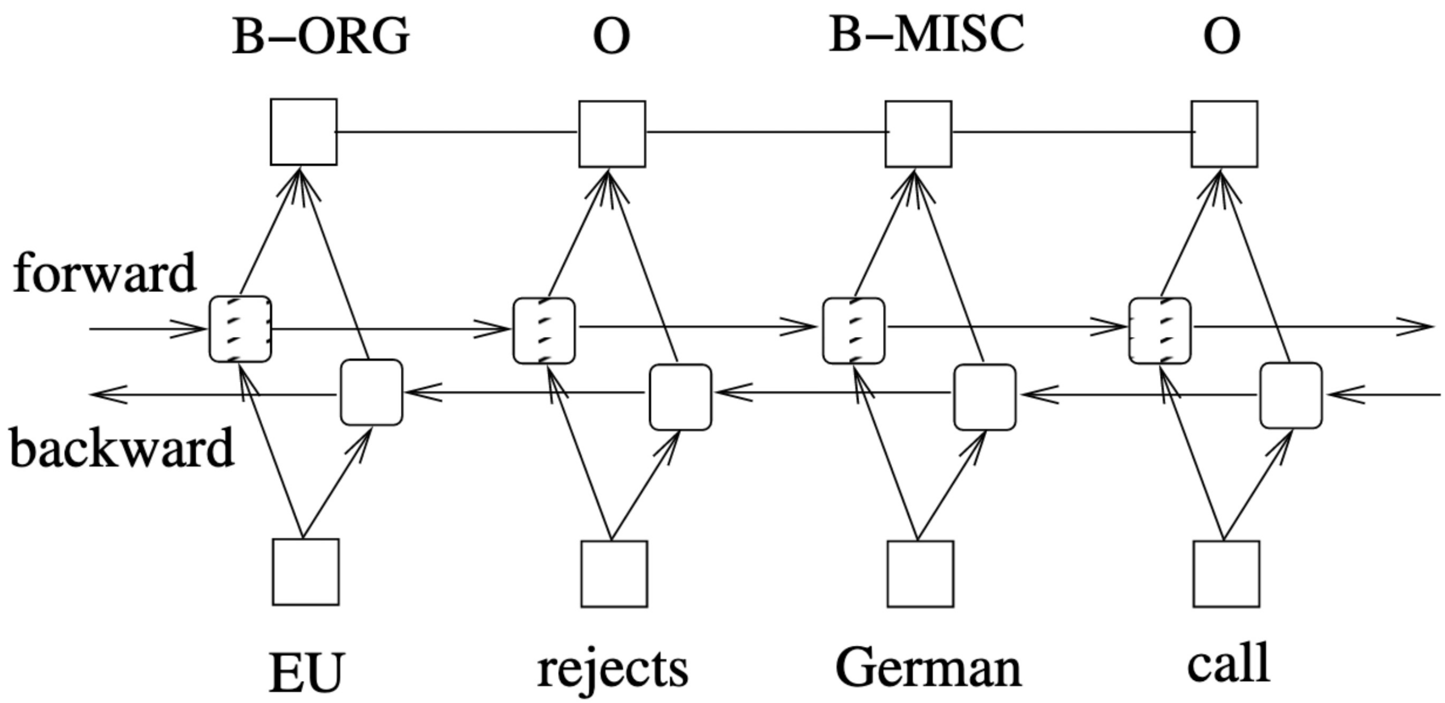
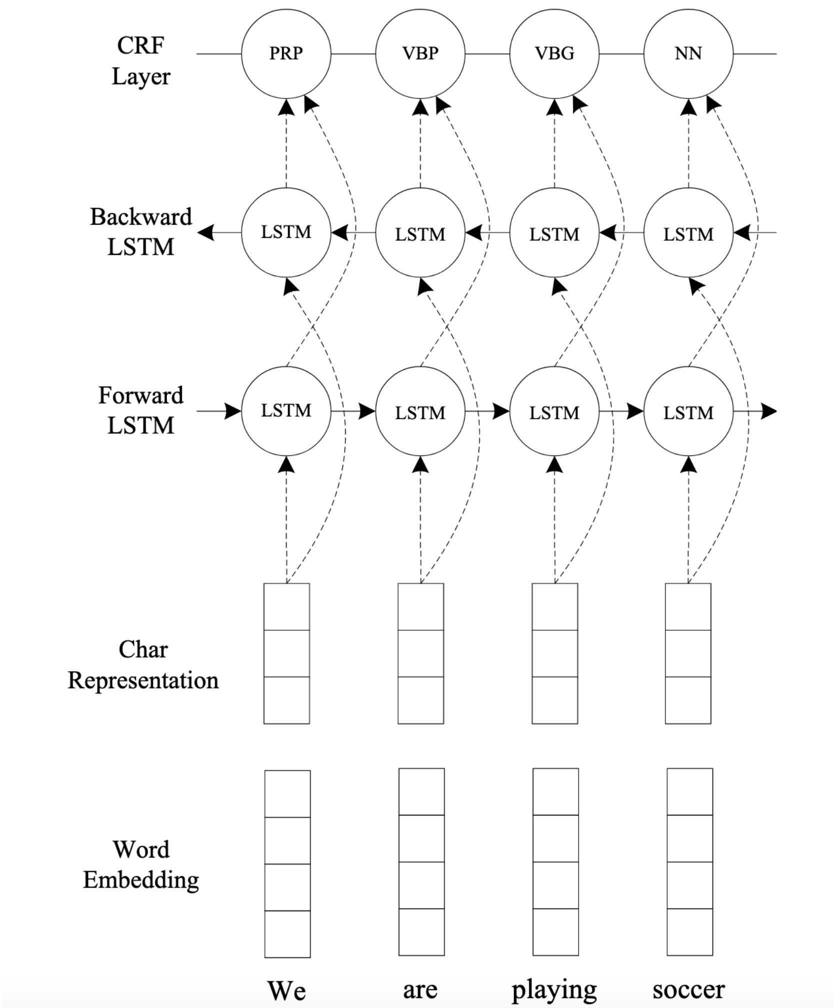


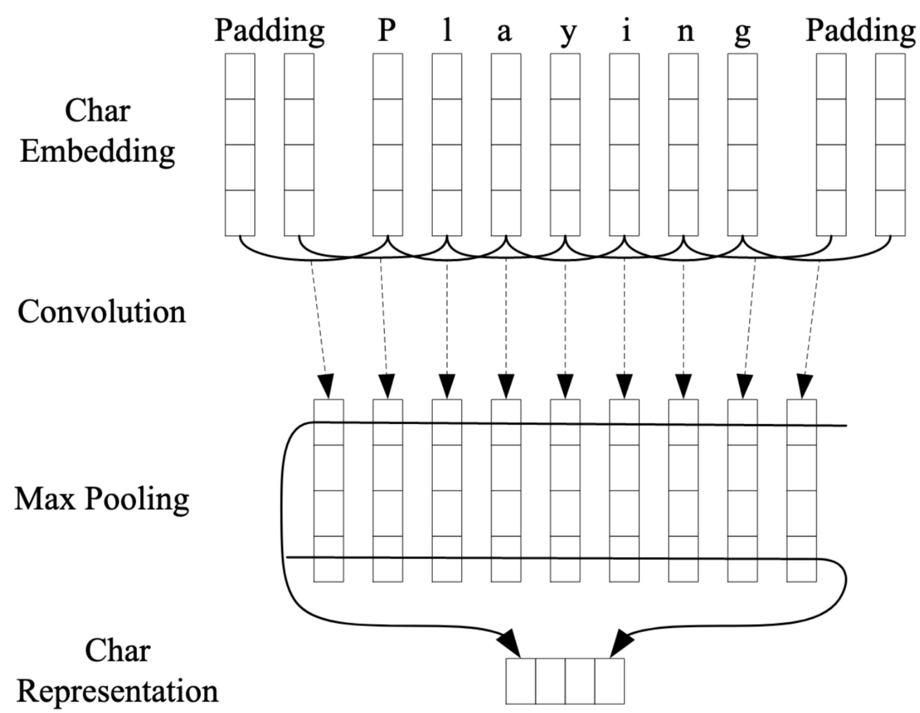
Figure 7: A BI-LSTM-CRF model.

$$s([x]_1^T, [i]_1^T, \tilde{\theta}) = \sum_{t=1}^{T} ([A]_{[i]_{t-1},[i]_t} + [f_\theta]_{[i]_t,t})$$

Huang et al. 2015, “Bidirectional LSTM-CRF Models for Sequence Tagging

Incorporating CRF





Ma and Hovy (2016), "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF"

Incorporating CRF

Layer	Hyper-parameter	POS	NER
CNN	window size	3	3
	number of filters	30	30
LSTM	state size	200	200
	initial state	0.0	0.0
	peepholes	no	no
Dropout	dropout rate	0.5	0.5
	batch size	10	10
	initial learning rate	0.01	0.015
	decay rate	0.05	0.05
	gradient clipping	5.0	5.0

	POS		NER					
	Dev	Test	Dev			Test		
Model	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	F1
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BRNN-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21