Missing values

- Oh no!

  ```
  TypeError: 'float' object is not iterable
  ```

- This means: you tried to concatenate a list and a number, but how did a number get into the ‘bow’ column?

- Pandas uses the special numeric value `NaN` (‘not a number’) to represent missing values

- read_csv interprets certain values as missing datapoints:

  ```
  '', '#N/A', '#N/A N/A', '#NA', '-1.#IND', '-1.#QNAN',
  '-NaN', '-nan', '1.#IND', '1.#QNAN', 'N/A', 'NA',
  'NULL', 'NaN', 'n/a', 'nan', 'null'
  ```

Missing values

- Solutions:
  - Skip `NaN` values when counting words:

    ```
    list(concat(df['bow'].dropna()))
    ```

  - Skip `NaN` processing when reading files:

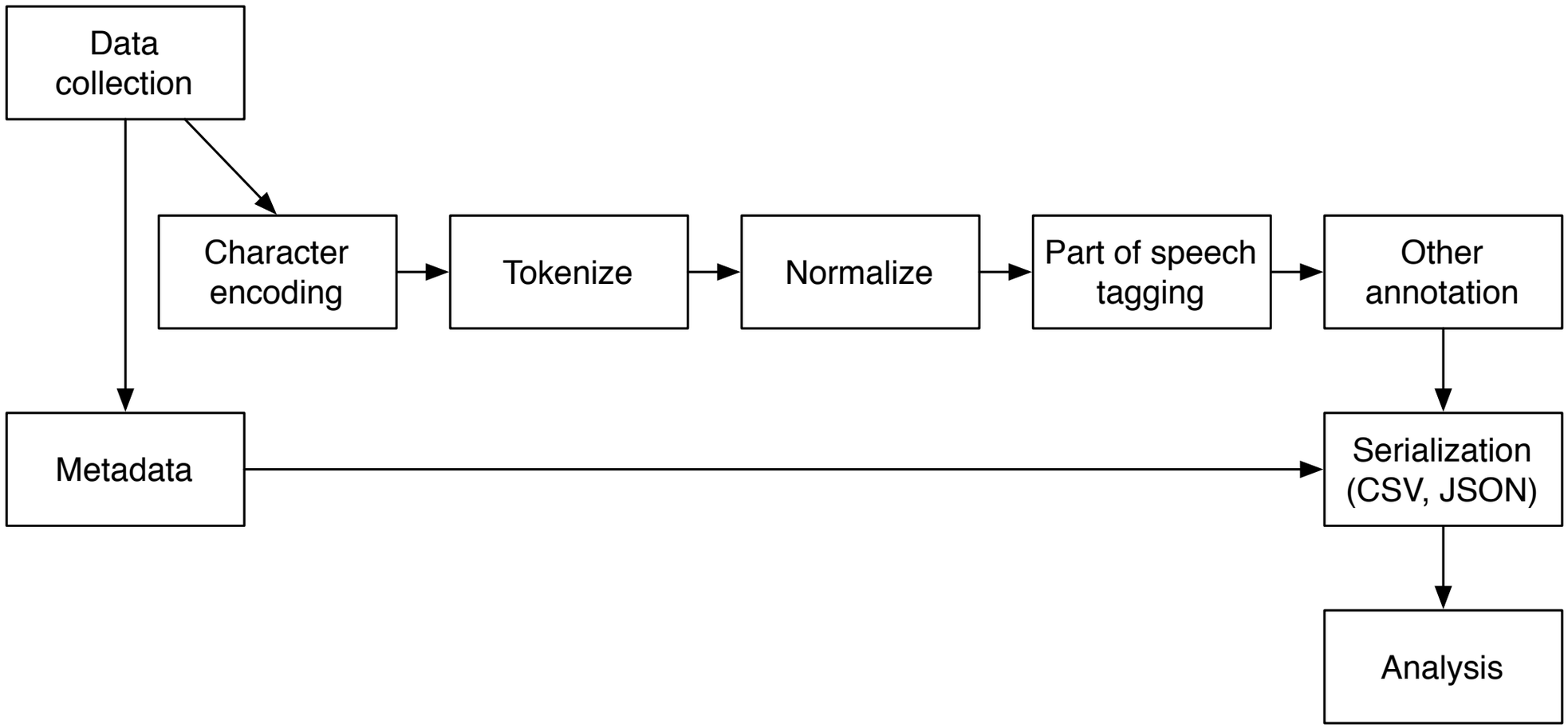
    ```
    pd.read_csv('file.csv', na_filter=False)
    ```

  - Warn about `NaN`s when reading files:

    ```
    pd.read_csv('file.csv', verbose=True)
    ```

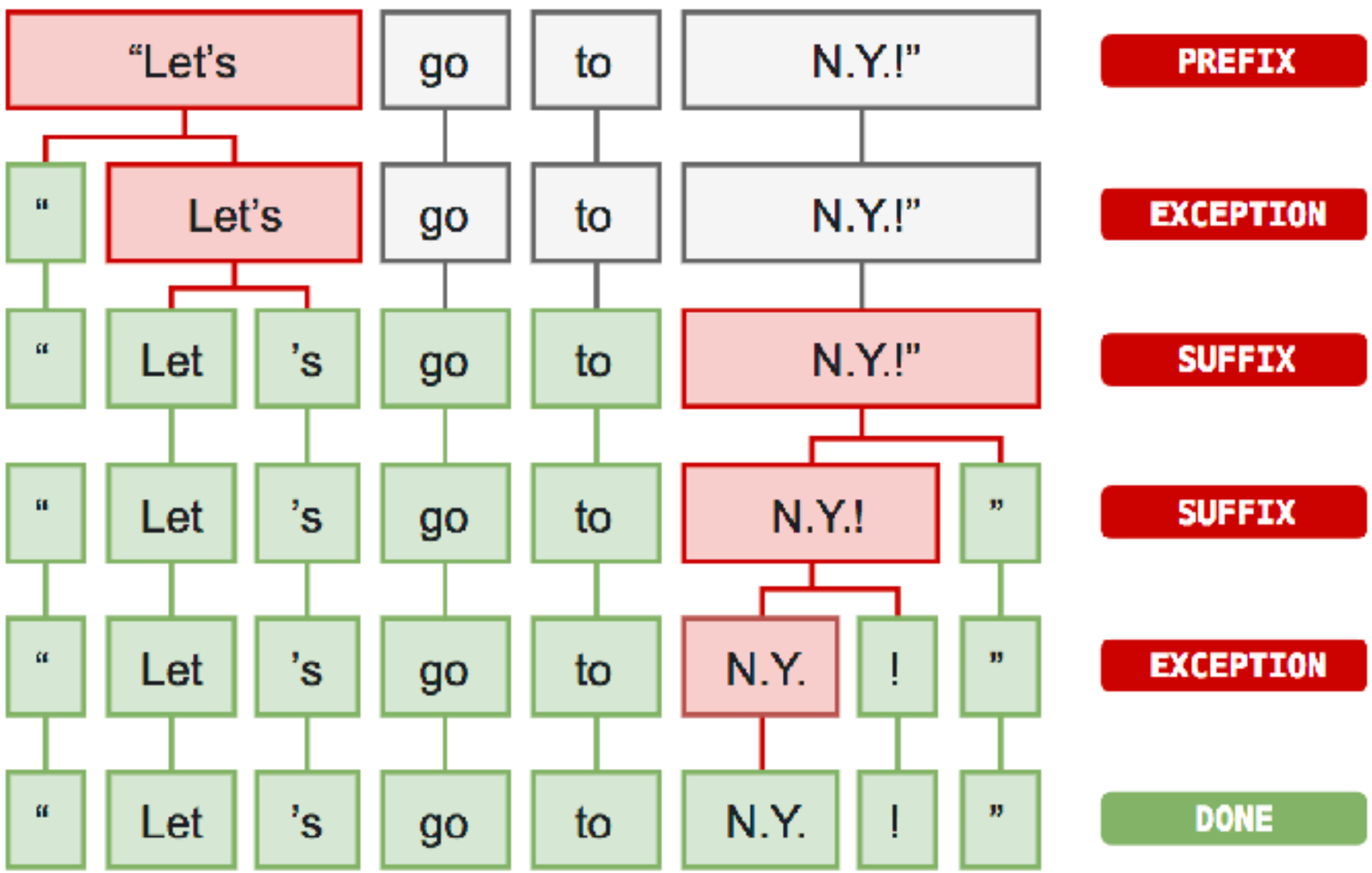
Text annotation

- Text processing often uses a ‘pipeline’ model (spacy)



PTB tokenization

- spacy tokenizer uses prefix, infix, suffix, and exception rules



Tagging

- Part-of-speech tagging assigns a grammatical category to tokens in a corpus
- Since words may potentially occur as more than one part of speech, tagging is a limited kind of disambiguation:

The	*representative*	*put*	*the*	*chairs*	*on*	*the*	*table*	*.*
DET	NOUN	VERB	DET	NOUN	PREP	DET	NOUN	PERIOD

- Tagging can be done by hand, automatically, or as a combination of the two.

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*			

Figure 10.1 Penn Treebank part-of-speech tags (including punctuation).

Stochastic tagging

- Stochastic taggers take advantage of uneven tag probabilities
- If we want to assign a sequence of tags $t_1, \ldots, t_n$ to a sequence of words $w_1 \ldots w_n$, we need to maximize:

$$P(t_1, \ldots, t_n \mid w_1, \ldots, w_n)$$

- We construct a probability model using some annotated data.
- Typically, the annotated data available is divided into **training**, **validation**, and **test** data.

Stochastic tagging

- The simplest statistical tagger picks the most frequent tag for a given word
- The probability of a tag given a word can be estimated from training data:

$$P(\text{tag}|\text{word}) = \frac{\#\text{ times word occurs with tag}}{\#\text{ times word occurs}}$$

- Depending on the text and the tagset, this tagger can give as high as 91% word accuracy
- Since it is so simple, it is usually considered the **baseline**
- The 'human ceiling' is about 98%

HMM taggers

- If we make a few simplifying assumptions, HMMs provide a tool for adding context:
  - a word's tag only depends on the previous word's tag (**Markov** property)
- From the training data, we need to collect probabilities for tag bigrams:

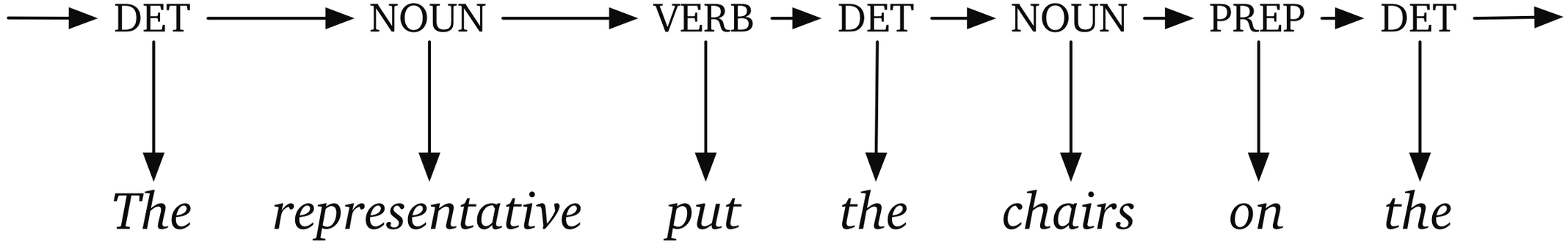
$$P(t_2|t_1) = \frac{C(t_1\, t_2)}{C(t_1)}$$

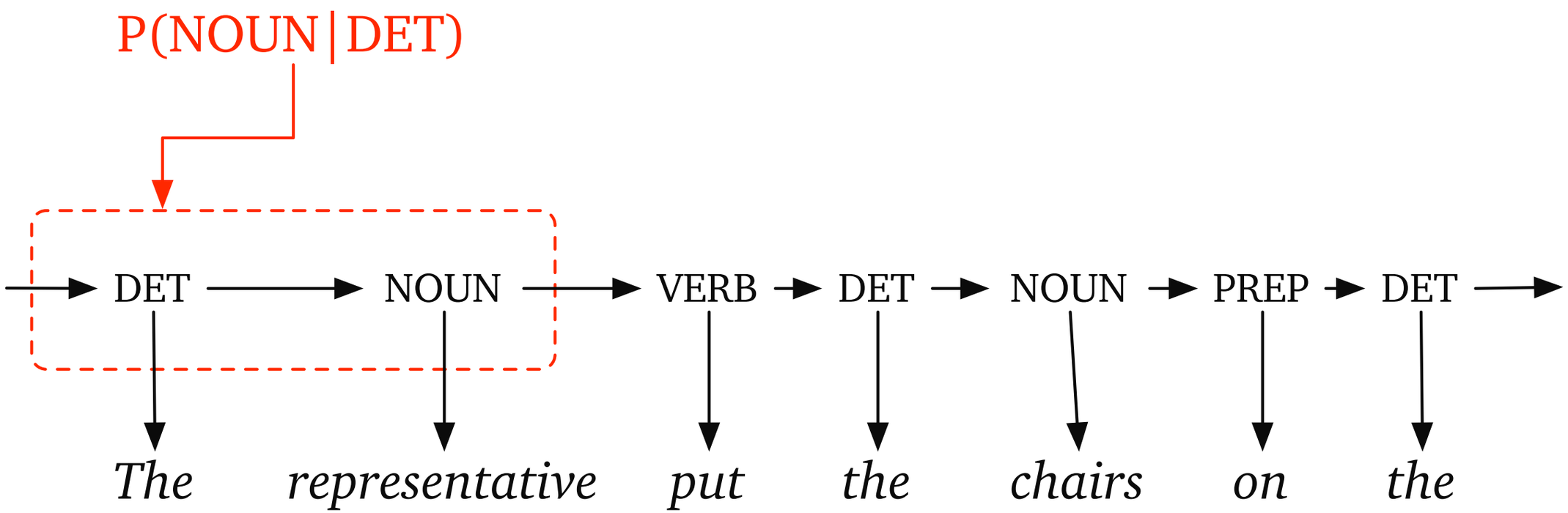
- And we need:

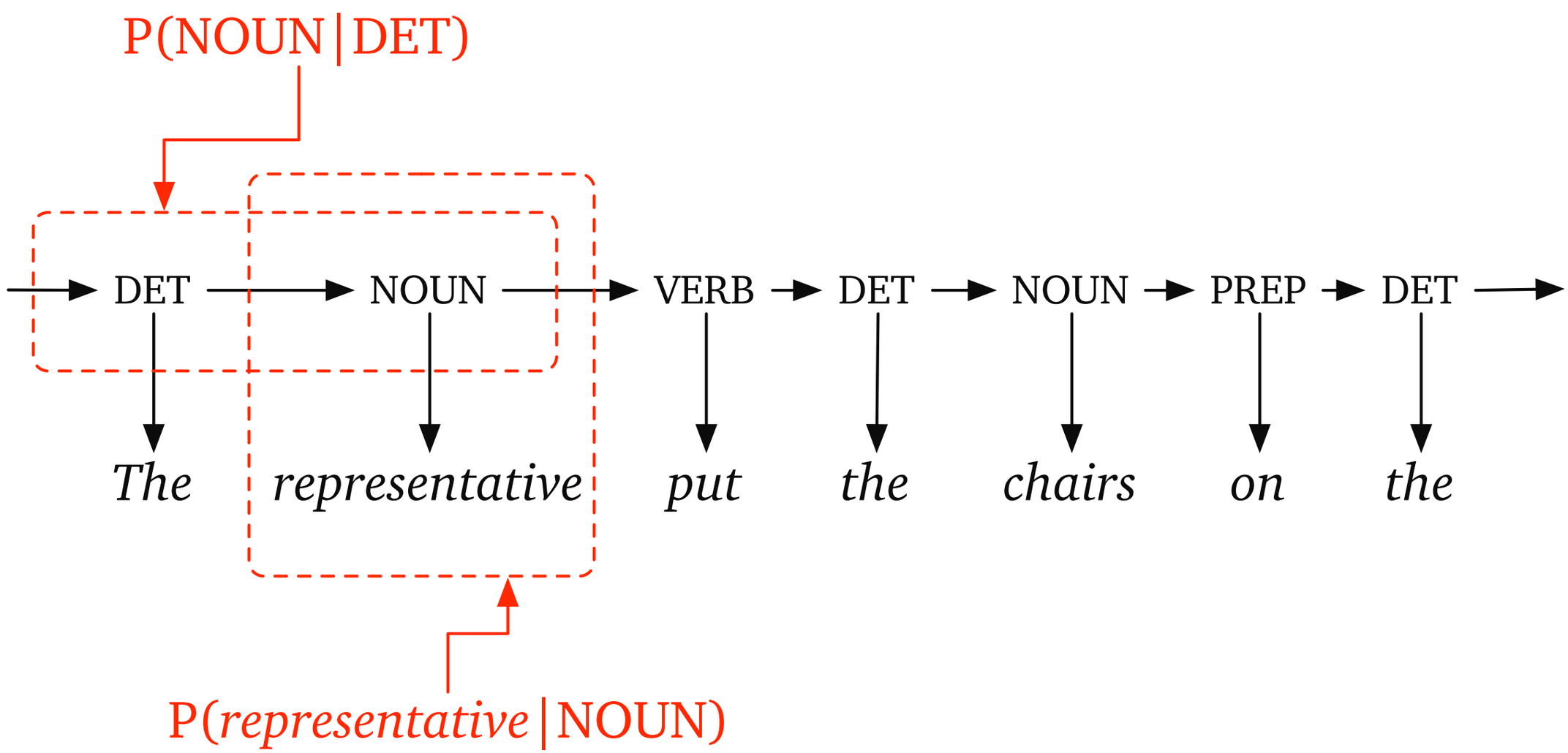
$$P(w|t) = \frac{C(w/t)}{C(t)}$$

HMM taggers

$$
\begin{aligned}
\hat{t}_1, \ldots, \hat{t}_n &= \operatorname*{argmax}_t P(t_1, \ldots, t_n | w_1, \ldots, w_n) \\
&= \operatorname*{argmax}_t \frac{P(t_1, \ldots, t_n)\, P(w_1, \ldots, w_n | t_1, \ldots, t_n)}{P(w_1, \ldots, w_n)} \\
&= \operatorname*{argmax}_t P(t_1, \ldots, t_n)\, P(w_1, \ldots, w_n | t_1, \ldots, t_n) \\
&\approx \operatorname*{argmax}_t \prod_i P(t_i | t_{i-1}) \prod_i P(w_i | t_i) \\
&= \operatorname*{argmax}_t \prod_i P(t_i | t_{i-1})\, P(w_i | t_i)
\end{aligned}
$$


DET
NOUN
VERB
DET
NOUN
PREP
DET
The
representative
put
the
chairs
on
the


P(NOUN|DET)
DET
NOUN
VERB
DET
NOUN
PREP
DET
The
representative
put
the
chairs
on
the


P(NOUN|DET)
DET
NOUN
VERB
DET
NOUN
PREP
DET
The
representative
put
the
chairs
on
the
P(representative|NOUN)

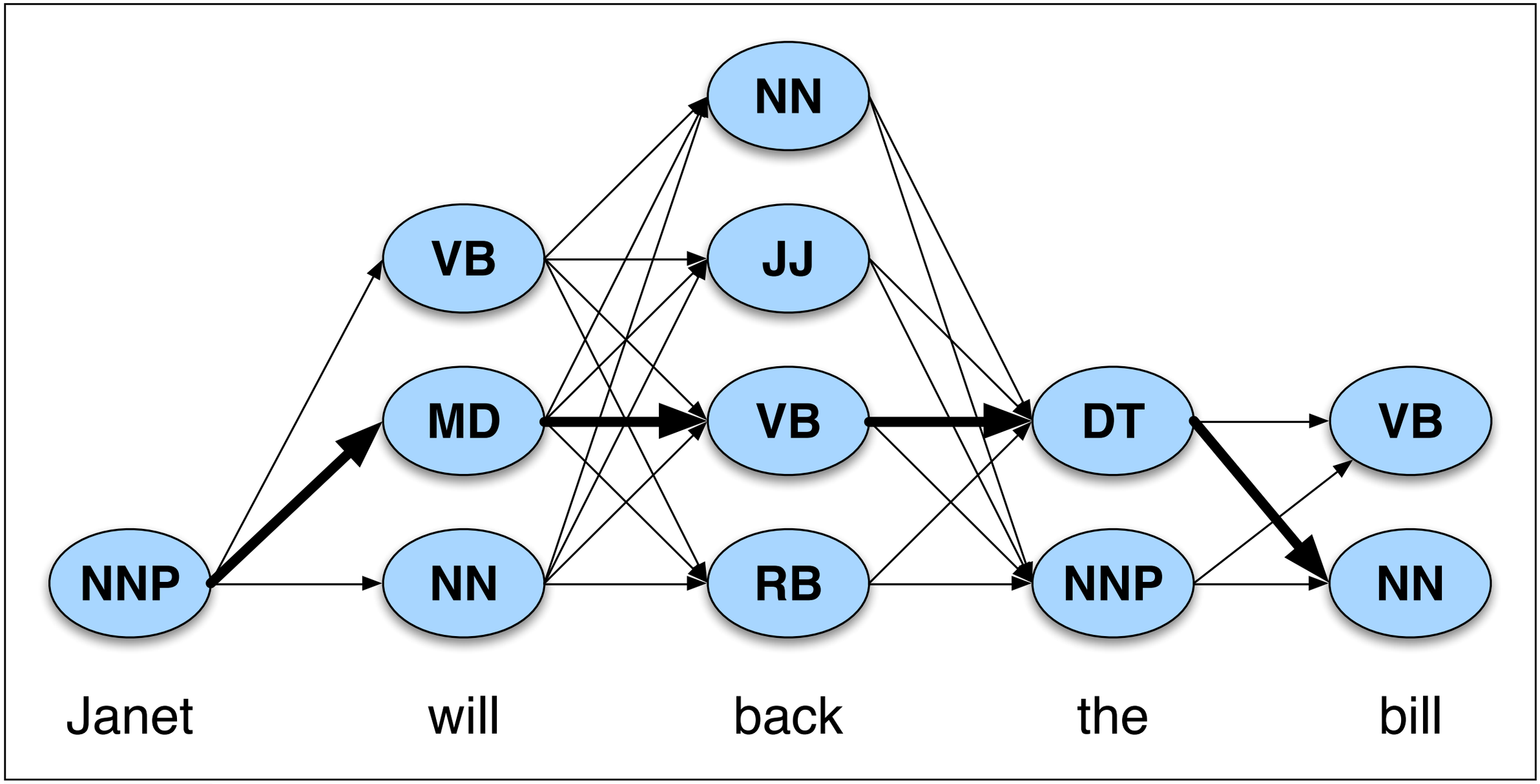


Figure 10.7 A schematic of the tagging task for the sample sentence, showing the ambiguities for each word and the correct tag sequence as the highlighted path through the hidden states.

HMM taggers

- Markov models can be extended to take more context to account (bigrams, trigrams, etc.)
- Problems with Markov taggers:
  - unknown words
  - long-distance relations
  - independence assumptions
- More sophisticated statistical models are needed.

HMM taggers

- Weischedel, et al.'s (1993) "tri-tag" model used trigram probabilities:

$$P(t_1, \ldots, t_n) = P(t_1) \times P(t_2|t_1) \times \prod_{i=3}^{n} P(t_i|t_{i-1}t_{i-2})$$

- For known words, $P(w_i|t_i)$ is estimated in the usual way
- For unknown words:

$$P(w_i|t_i) \quad = \quad P(\text{unknown}|t_i) \times P(\text{capital}|t_i) \times P(\text{hyphen}|t_i) \times \prod_j P(\text{ending}_j|t_i)$$

- Works pretty well (85% unknown word accuracy), despite dubious independence assumptions

HMM taggers

- Brants (2000) extended this model to be less language-specific
  - For each suffix (up to 10 letters), calculate $P(t_i|l_{n-i+1},\ldots,l_n)$ based on words with occur ten or fewer times in the training data
  - Combine suffix probabilities of different length using backoff smoothing
  - Capitalization also gets factored in
  - Compute $P(w_i|t_i)$ from $P(t_i|w_i)$ using Bayes' rule

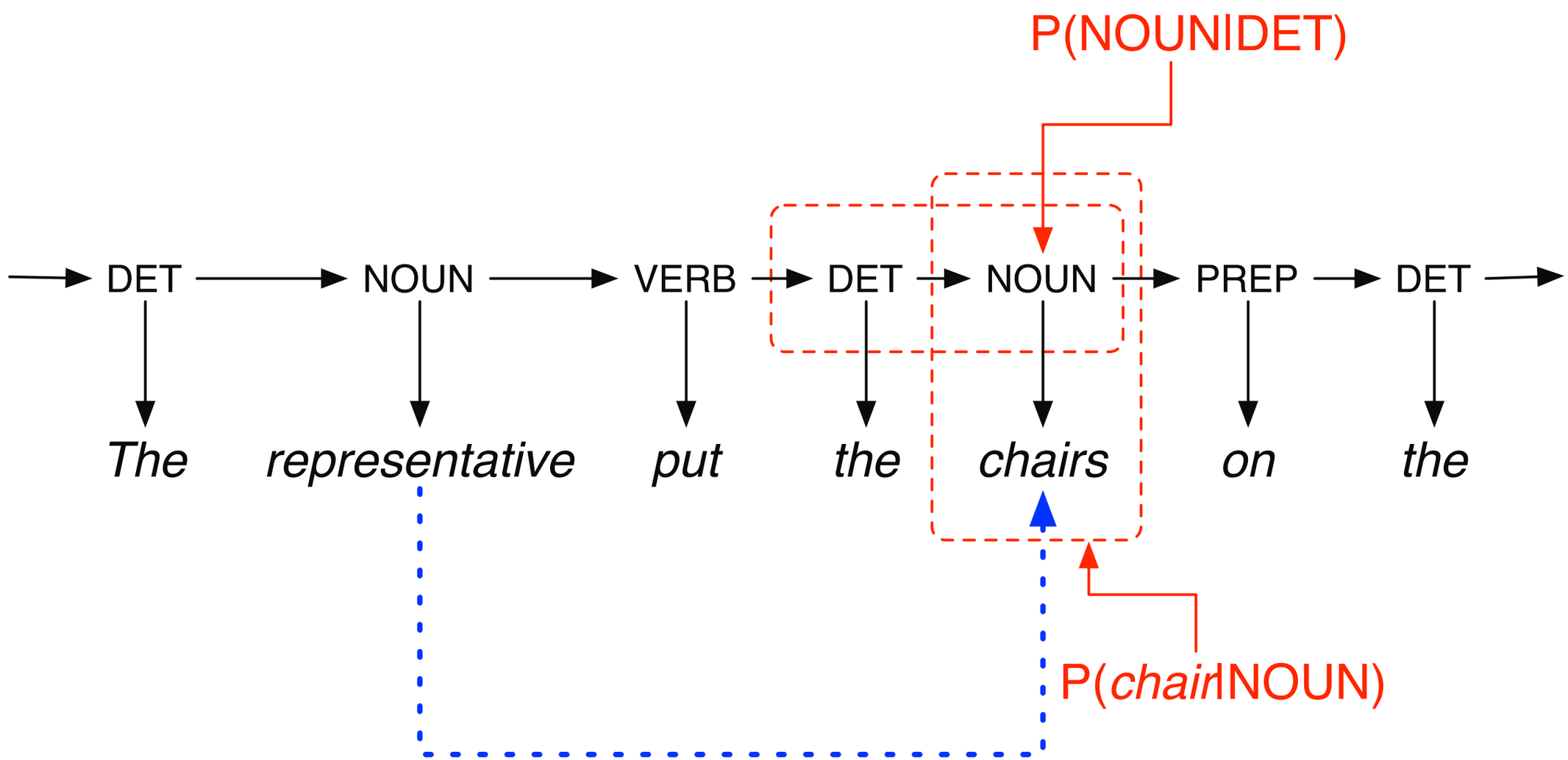
Language models

- Grammar-based models incorporate information about derivations:

$$P(w_1, \ldots, w_n) = \sum_i P(w_1, \ldots, w_n | d_i) P(d_i)$$

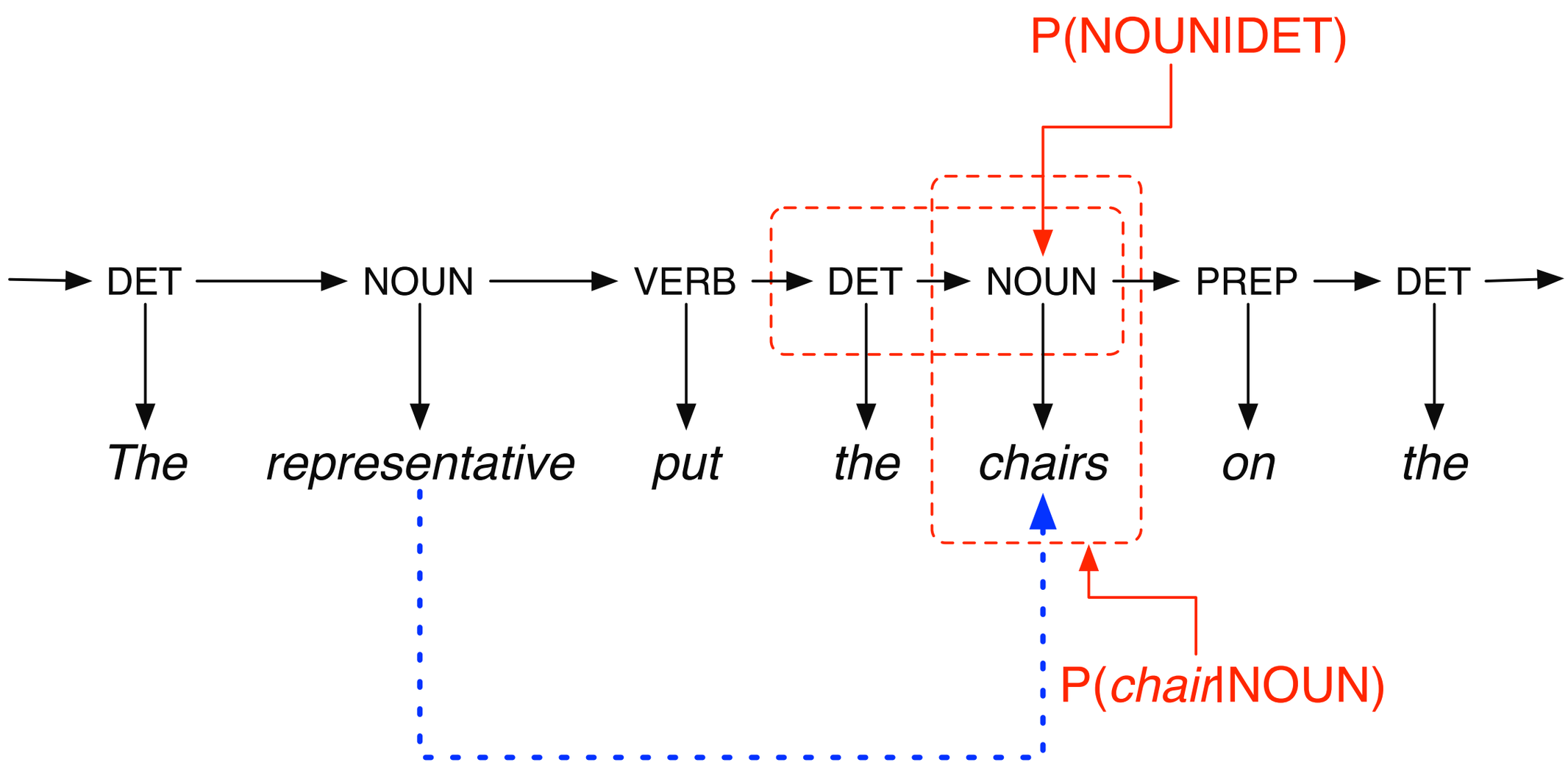
- **Trigger** models use long distance features. For example, if the word *stock* appears in a sentence, the word *shares* is more likely to occur in the next four to ten words
- Adding trigger features complicates independence assumptions.
- Is this valid?

$$P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1}) \prod_j P(w_i | t_j)$$


P(NOUN|DET)
DET
NOUN
VERB
DET
NOUN
PREP
DET
The
representative
put
the
chairs
on
the
P(chair|NOUN)

Logistic regression

- Logistic regression ('maxent') provides a general strategy for choosing distributions
- What distribution do we want for trigger *n*-gram models?
- We represent each training instance as a feature vector *x*



DET, NOUN (2)
NOUN, VERB (1)
VERB, DET (1)
NOUN, PREP (1)
PREP, DET (1)

DET, The (1)
NOUN, representative (1)
VERB, put (1)
DET, the (1)
NOUN, chairs (1)
PREP, on (1)
DET, the (1)

representative . . . chairs (1)

Logistic regression

- We have a bunch of statistics (e.g., *n*-gram frequencies, word/trigger co-occurrences) that we want to model
- We want to select a model which satisfies the empirical constraints $f(w)$ imposed by the training data and minimizes the entropy:

$$P(w|h) = \frac{\exp\left(\sum_i \lambda_i f_i(h, w)\right)}{\sum_z \exp\left(\sum_i \lambda_i f_i(h, z)\right)}$$

- There are general convex optimization procedures for selecting the parameter vector λ

MEMMs

- MaxEnt models can use arbitrary properties of the data to compute probabilities
- Maximum Entropy Markov Models (MEMMs) are based on transition and emission features

$$P(t|w,t') = \frac{\exp\left(\sum_i \lambda_i f_i(t,w,t')\right)}{\sum_z \exp\left(\sum_i \lambda_i f_i(z,w,t')\right)}$$

where features look like

$$f_{12}(t,w,t') = \begin{cases} 1 & \text{if } w \text{ is } \textit{singing } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

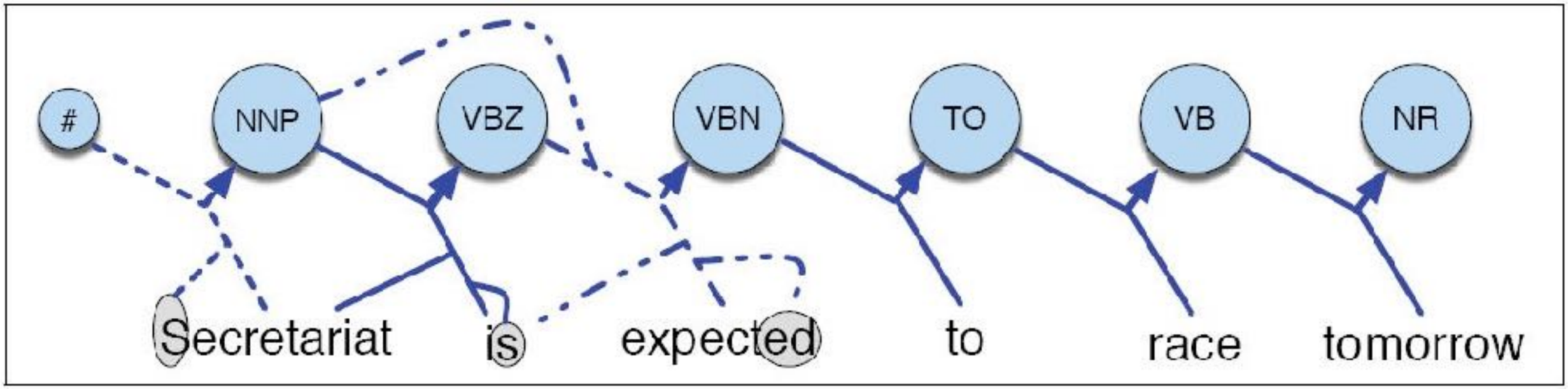
$$f_{34}(t,w,t') = \begin{cases} 1 & \text{if } t' = \texttt{AT} \text{ and } t = \texttt{NN} \\ 0 & \text{otherwise} \end{cases}$$

MaxEnt taggers

- Because MaxEnt models can use arbitrary properties, we aren't limited to just transition and emission probabilities (Ratnaparki 1996)
- We can incorporate properties like capitalization, hyphenation, and suffixes to the model directly (cf. *tri-tag* model)

$$f_{243}(t, w, t') = \begin{cases} 1 & \text{if } w \text{ ends in } \textit{-ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

- As long as the features are local, we can use the Viterbi algorithm for finding the most likely tag sequence


#
NNP
VBZ
VBN
TO
VB
NR
Secretariat
is
expected
to
race
tomorrow

spaCy

- spaCy is a cython implementation of an NLP pipeline
- Input is a unicode string, output a Doc (i.e., a sequence of Tokens)
- Tokens contain linguistic information about tokens
  - orth_, lemma_, lower_, norm_, pos_, tag_,
  - is_alpha, is_ascii, is_bracket, is_digit, is_left_punct, is_lower, is_oov, is_punct, is_quote, is_right_punct, is_space, is_stop, is_title, is_upper
  - like_email, like_num, like_url

Beyond tagging

- Tagging is an instance of the more general problem of **sequence labeling**
- Other interesting problems can be solved in the same way
- **Word sense disambiguation** is a kind of semantic tagging:
  - The representative put the chair on the table.
  - *chair* = *piece of furniture* or *committee leader*?
  - *put on the table* = *place on furniture* or *make a proposal*?
- WSD can be performed using the same methods as tagging, but much more difficult

Named entities

- [$_{\text{PER}}$ Wolff] , currently a journalist in [$_{\text{LOC}}$ Argentina] , played with [$_{\text{PER}}$ Del Bosque] in the final years of the seventies in [$_{\text{ORG}}$ Real Madrid] .
- Labeled bracketing is a kind of tagging: B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG, O
- Challenges:
  - Many unknown words: *Bayan Ölgiy*, *Nambaryn Enkhbayar*
  - Ambiguity: *Washington* can be a person, location, or organization

[ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said.

Words	IOB Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O
matched	O	O
the	O	O
move	O	O
,	O	O
spokesman	O	O
Tim	B-PER	I-PER
Wagner	I-PER	I-PER
said	O	O
.	O	O

Figure 21.4 Named entity tagging as a sequence model, showing IOB and IO encodings.

identity of w_i
identity of neighboring words
part of speech of w_i
part of speech of neighboring words
base-phrase syntactic chunk label of w_i and neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i is all upper case
word shape of w_i
word shape of neighboring words
short word shape of w_i
short word shape of neighboring words
presence of hyphen

Figure 21.5 Features commonly used in training named entity recognition systems.

Word	POS	Chunk	Short shape	Label
American	NNP	B-NP	Xx	B-ORG
Airlines	NNPS	I-NP	Xx	I-ORG
,	,	O	,	O
a	DT	B-NP	x	O
unit	NN	I-NP	x	O
of	IN	B-PP	x	O
AMR	NNP	B-NP	X	B-ORG
Corp.	NNP	I-NP	Xx.	I-ORG
,	,	O	,	O
immediately	RB	B-ADVP	x	O
matched	VBD	B-VP	x	O
the	DT	B-NP	x	O
move	NN	I-NP	x	O
,	,	O	,	O
spokesman	NN	B-NP	x	O
Tim	NNP	I-NP	Xx	B-PER
Wagner	NNP	I-NP	Xx	I-PER
said	VBD	B-VP	x	O
.	,	O	.	O

Figure 21.6 Word-by-word feature encoding for NER.

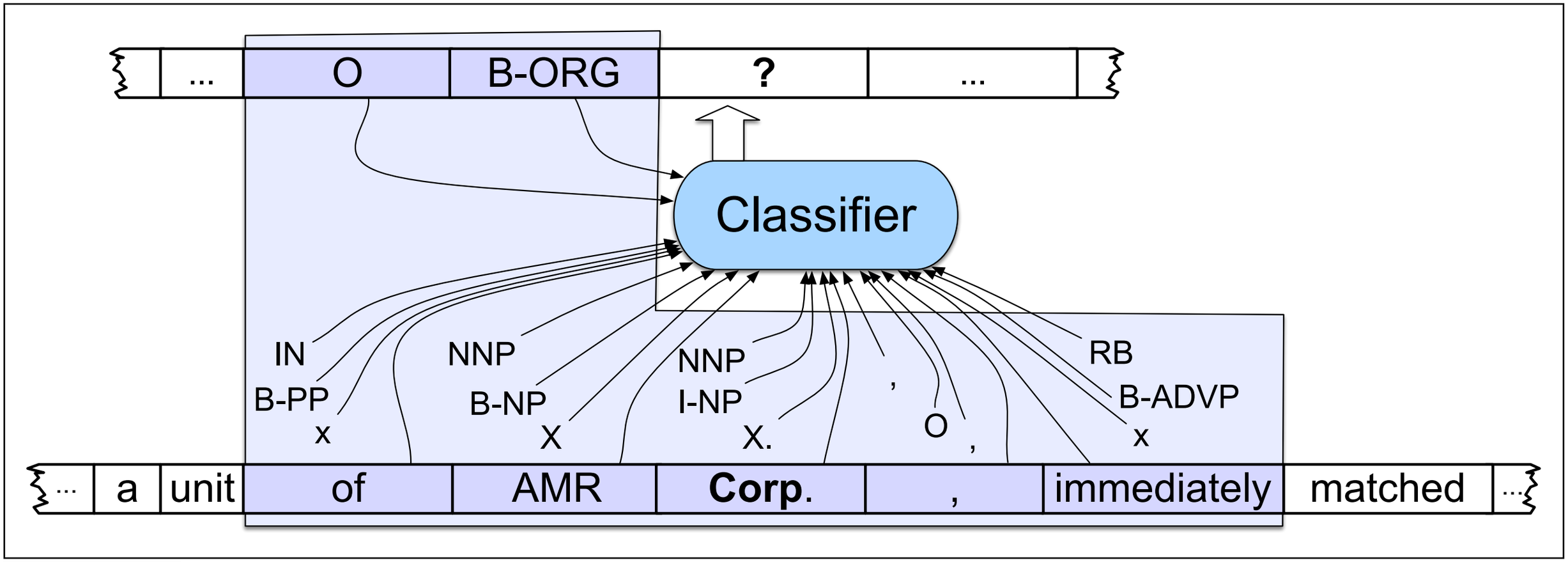


Figure 21.7 Named entity recognition as sequence labeling. The features available to the classifier during training and classification are those in the boxed area.

`PERSON`	People, including fictional.
`NORP`	Nationalities or religious or political groups.
`FACILITY`	Buildings, airports, highways, bridges, etc.
`ORG`	Companies, agencies, institutions, etc.
`GPE`	Countries, cities, states.
`LOC`	Non-GPE locations, mountain ranges, bodies of water.
`PRODUCT`	Objects, vehicles, foods, etc. (Not services.)
`EVENT`	Named hurricanes, battles, wars, sports events, etc.
`WORK_OF_ART`	Titles of books, songs, etc.
`LAW`	Named documents made into laws.
`LANGUAGE`	Any named language.
`DATE`	Absolute or relative dates or periods.
`TIME`	Times smaller than a day.
`PERCENT`	Percentage, including "%".
`MONEY`	Monetary values, including unit.
`QUANTITY`	Measurements, as of weight or distance.
`ORDINAL`	"first", "second", etc.
`CARDINAL`	Numerals that do not fall under another type.