# Sequence Labeling for Parts of Speech and Name Entities

Berlin Chen

Department of Computer Science & Information Engineering
National Taiwan Normal University

References:

1. *Speech and Language Processing*, 1st & 3rd, both Chapter 8, & Teaching Material
2. *Foundations of Statistical Natural Language Processing*, Chapter 10

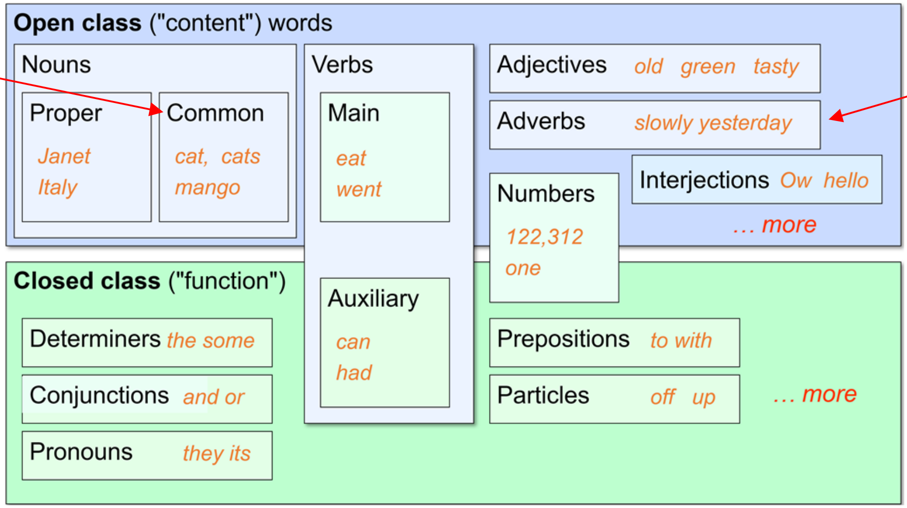
# Prologue (1/3)

- Eight parts of speech attributed to Dionysius Thrax of Alexandria (*c*. 100 B.C.)
  - noun, verb, pronoun, preposition, adverb, conjunction, participle, article
  - These categories are relevant for NLP today
- The earliest implemented part-of-speech assignment algorithm may have been part of the parser in Zellig Harris's (1962) Transformations and Discourse Analysis Project (TDAP)
- Closed class words
  - Have relatively fixed membership
  - Usually **function** words: short, frequent words with grammatical function
    - determiners: *a, an, the*
    - pronouns: *she, he, I*
    - prepositions: *on, under, over, near, by, …*

# Prologue (2/3)

- Open class words
  - Usually **content** words: Nouns, Verbs, Adjectives, Adverbs
    - Plus interjections: *oh, ouch, uh-huh, yes, hello*
  - New nouns and verbs like *iPhone* or *to fax*

count nouns &
mass nouns



directional
degree
manner
temporal

adverbs

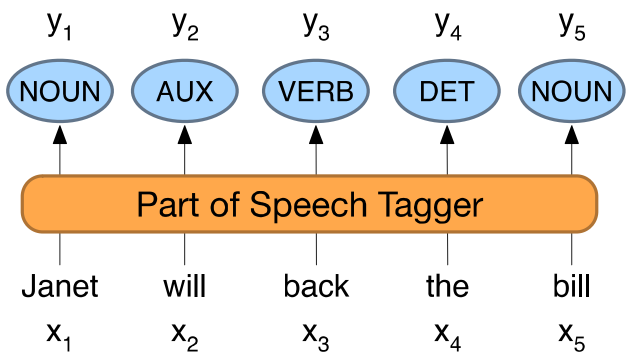
- Proper names (viz. named entities) are another important and anciently studied linguistic category (e.g., *New York City*, *Stanford University*)

# Prologue (3/3)

- Applications of POS information
  - A word's part-of-speech can tell us something about how the word is pronounced
    - Beneficial for speech synthesis/text to speech (TTS)
    - POS taggers are also used in advanced language models for automatic speech recognition (ASR) such as class-based
    - POS can also be used in stemming for informational retrieval (IR)

- A side note: writing Issues in English (for example, proper nouns)
  - Proper nouns, like *Regina*, *Colorado*, and *IBM*, are names of specific persons or entities
  - In English, proper nouns generally are not preceded by articles (e.g. *the book is upstairs*, but *Regina is upstairs*)
  - In written English, proper nouns are usually capitalized

# POS Tagging: Task Definition



- Tagging (part-of-speech tagging)
  - To assign a part-of-speech (POS) or other lexical class marker to each word in a text (e.g., sentence or document)
    - Decide whether each word is a noun, verb, adjective, or whatever (words often have more than one POS)

The/AT representative/NN put/VBD chairs/NNS on/IN the/AT table/NN ✓

Or

The/AT representative/JJ put/NN chairs/VBZ on/IN the/AT table/NN

  - An intermediate layer of representation of syntactic structure
    - When tagging is compared with syntactic parsing
- Accuracies across different languages are all about 97% (why?)
  - No matter the algorithms: HMMs, CRFs, BERT perform similarly)
  - This 97% number is also about the human performance on this task, at least for English (Manning, 2011)

Tagging can be viewed as a kind of syntactic disambiguation

# Tag ambiguity in the Brown and WSJ corpora

| **Types:** | **WSJ** | **Brown** |
|---|---|---|
| **Unambiguous** (1 tag) | 44,432 (**86%**) | 45,799 (**85%**) |
| **Ambiguous** (2+ tags) | 7,025 (**14%**) | 8,050 (**15%**) |
| Tokens: | | |
| **Unambiguous** (1 tag) | 577,421 (**45%**) | 384,349 (**33%**) |
| **Ambiguous** (2+ tags) | 711,780 (**55%**) | 786,646 (**67%**) |

- Most word types (85-86%) are unambiguous
- But the ambiguous words, though accounting for only 14-15% of the vocabulary, are very common, and 55-67% of word tokens in running text are ambiguous
- Between 96% and 97% of tokens are disambiguated correctly by the most successful tagging approaches

## Introduction

- Parts-of-speech
  - Known as POS, word classes, lexical tags, morphology classes
- Tag sets
  - Penn Treebank : 45 word classes used (Francis, 1979)
    - Penn Treebank is a parsed corpus
  - Brown corpus: 87 word classes used (Marcus et al., 1993)
  - ….

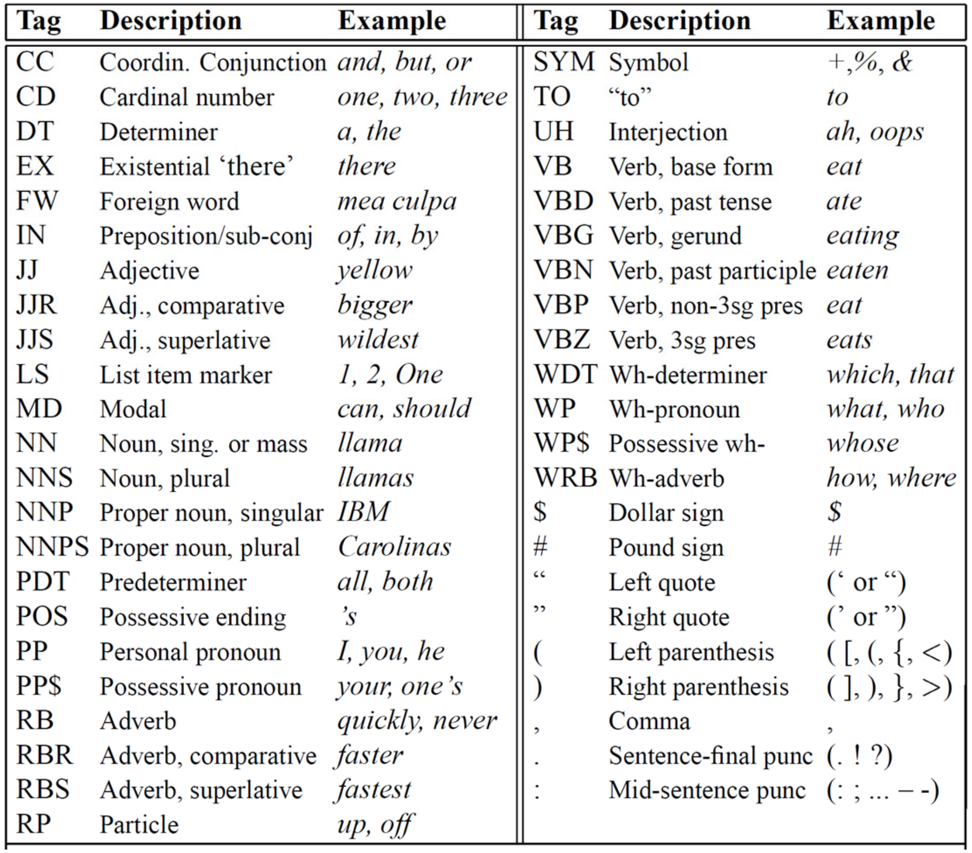
Which tagset to use for a particular application depends, of course, on how much information the application needs.

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

# The Penn Treebank POS Tag Set

- Penn Treebank Part-of-Speech Tags (including punctuation)

45 tags
(Marcus *et al.*, 1993)

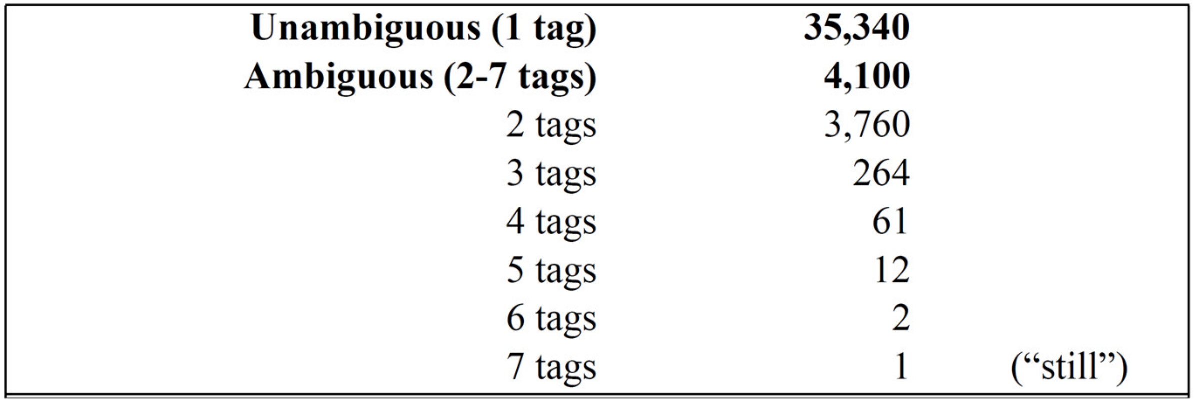


| Tag | Description | Example | Tag | Description | Example |
|---|---|---|---|---|---|
| CC | Coordin. Conjunction | *and, but, or* | SYM | Symbol | +,%, & |
| CD | Cardinal number | *one, two, three* | TO | “to” | *to* |
| DT | Determiner | *a, the* | UH | Interjection | *ah, oops* |
| EX | Existential ‘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, singular | *IBM* | $ | Dollar sign | *$* |
| NNPS | Proper noun, plural | *Carolinas* | # | Pound sign | # |
| PDT | Predeterminer | *all, both* | “ | Left quote | (‘ or “) |
| POS | Possessive ending | *’s* | ” | Right quote | (’ or ”) |
| PP | Personal pronoun | *I, you, he* | ( | Left parenthesis | ( [, (, {, <) |
| PP$ | 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* | | | |

The Penn Treebank tagset was culled from the original 87-tag tagset for the Brown corpus.

# Disambiguation

- Resolve the ambiguities and choose the proper tags for the given context
- Most English words are unambiguous (have only one tag) but many of the most common words are ambiguous
  - E.g.: "*can*" can be an auxiliary verb, main verb or a noun
  - E.g.: statistics of Brown corpus



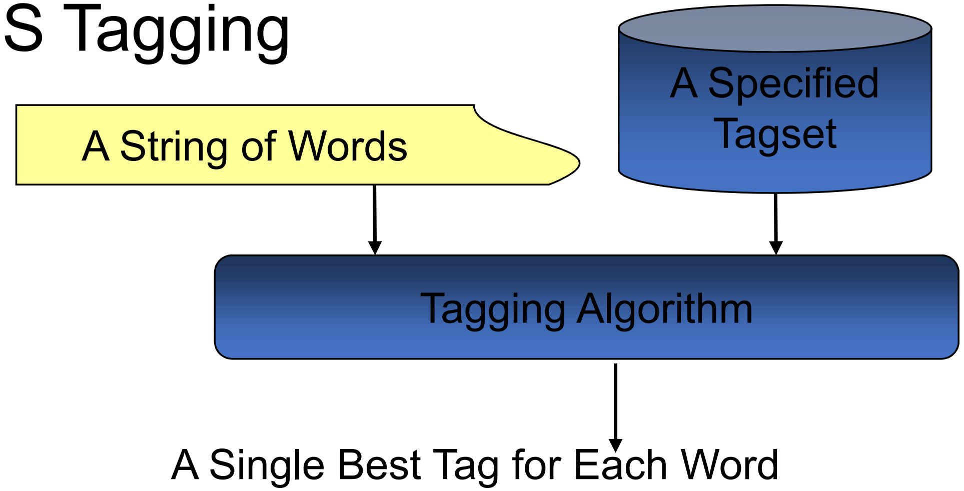
| | | |
|---|---|---|
| **Unambiguous (1 tag)** | **35,340** | |
| **Ambiguous (2-7 tags)** | **4,100** | |
| 2 tags | 3,760 | |
| 3 tags | 264 | |
| 4 tags | 61 | |
| 5 tags | 12 | |
| 6 tags | 2 | |
| 7 tags | 1 | ("still") |

The number of word types in Brown corpus by degree of ambiguity (after DeRose (1988)).

- 11.5% word types are ambiguous
- But 40% tokens are ambiguous
- However, the probabilities of tags associated a word are not equal
- many ambiguous tokens are easy to disambiguate

$$P(t_1|w) \neq P(t_2|w) \neq \cdots$$

# Process of POS Tagging



| VB | DT | NN | . |
|---|---|---|---|
| Book | that | flight | . |

| VBZ | DT | NN | VB | NN | ? |
|---|---|---|---|---|---|
| Does | that | flight | serve | dinner | ? |

the ATIS corpus of dialogues about air-travel reservations

**Two information sources used:**

- Syntagmatic information (looking at information about tag sequences)
- Lexical information (predicting a tag based on the word concerned)

Syntagmatic: denoting the relationship between two or more linguistic units used sequentially to make well-formed structures.

# POS Tagging Algorithms (1/2)

## Fall into One of Two Classes

- Rule-based Tagger
  - Involve a large database of handcrafted disambiguation rules
    - E.g. a rule specifies that an ambiguous word is a noun rather than a verb if it follows a determiner
    - **ENGTWOL**: a simple rule-based tagger based on the **constraint grammar** architecture of Karlsson *et al.(*1995)

"a new play"

*P*(NN|JJ) ≈ 0.45

*P*(VBP|JJ) ≈ 0.0005

- Stochastic/Probabilistic Tagger
  - Also called model-based tagger
  - Use a training corpus to compute the probability of a given word having a given context
  - E.g.: *the HMM tagger* and *CRF tagger* choose the best tag for a word in a given context

    (HMM maximizes the product of **word likelihood** and **tag sequence probability**)

# POS Tagging Algorithms (2/2)

- Transformation-based/Brill Tagger
  - A hybrid approach
  - Like rule-based approach, determine the tag of an ambiguous word based on rules
  - Like stochastic approach, the rules are automatically induced from previous tagged training corpus with the machine learning technique
    - Supervised learning

# Rule-based POS Tagging (1/3)

- Two-stage architecture
  - **First stage**: Use a dictionary to assign each word a list of potential parts-of-speech
  - **Second stage**: Use large lists of hand-written disambiguation rules to winnow down (精選) this list to a single part-of-speech for each word

Pavlov had shown that salivation …

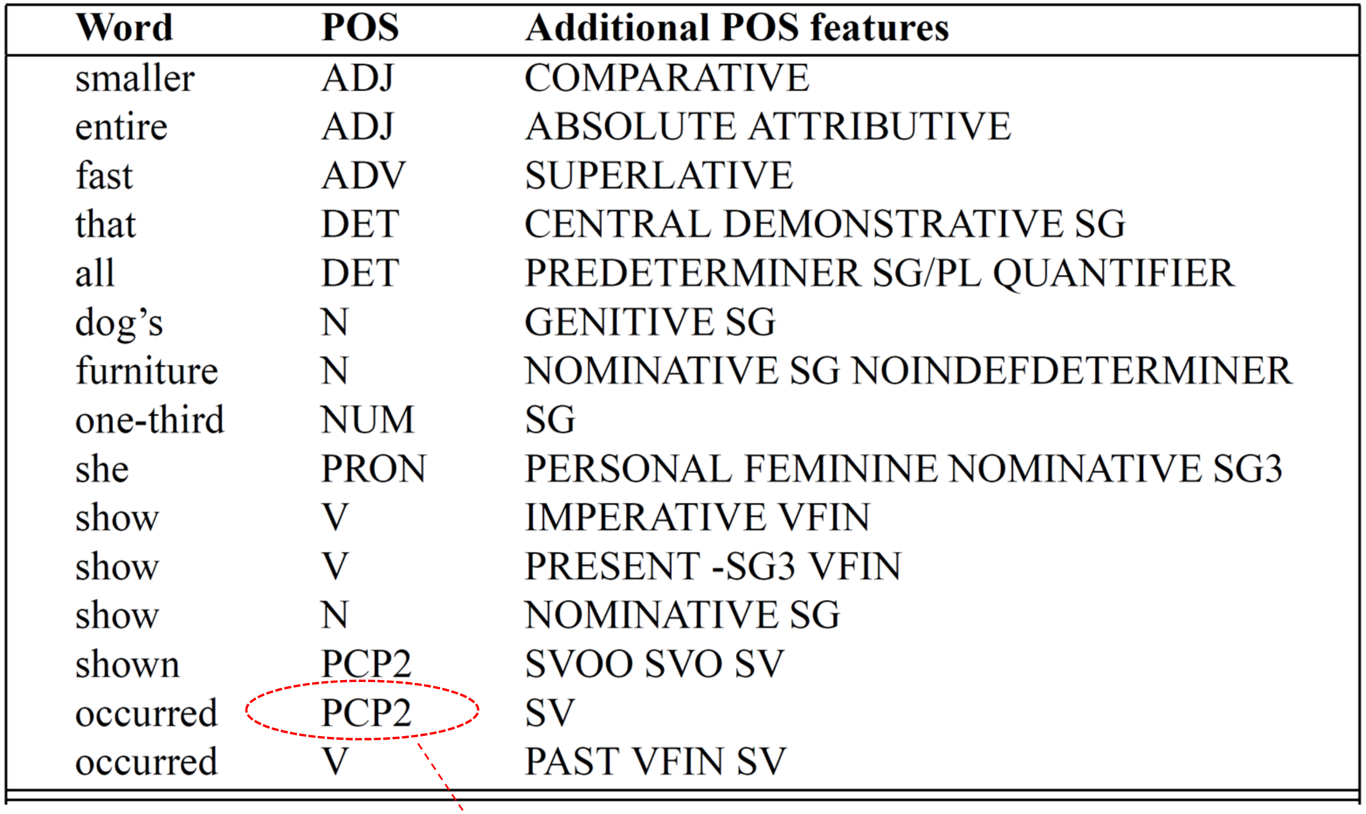
| | | |
|---|---|---|
| Pavlov | **PAVLOV N NOM SG PROPER** | |
| had | **HAVE V PAST VFIN SVO** | (preterit) |
| | HAVE PCP2 SVO | (past participle) |
| shown | **SHOW PCP2 SVOO SVO SV** | |
| that | ADV | |
| | PRON DEM SG | |
| | DET CENTRAL DEM SG | |
| | **CS** | (complementizer) |
| salivation | **N NOM SG** | |

*An example for*
*The ENGTOWL tagger*
*(Voutilainen, 1995)*

*A set of 1,100 constraints can be applied to the input sentence*

# Rule-based POS Tagging (2/3)

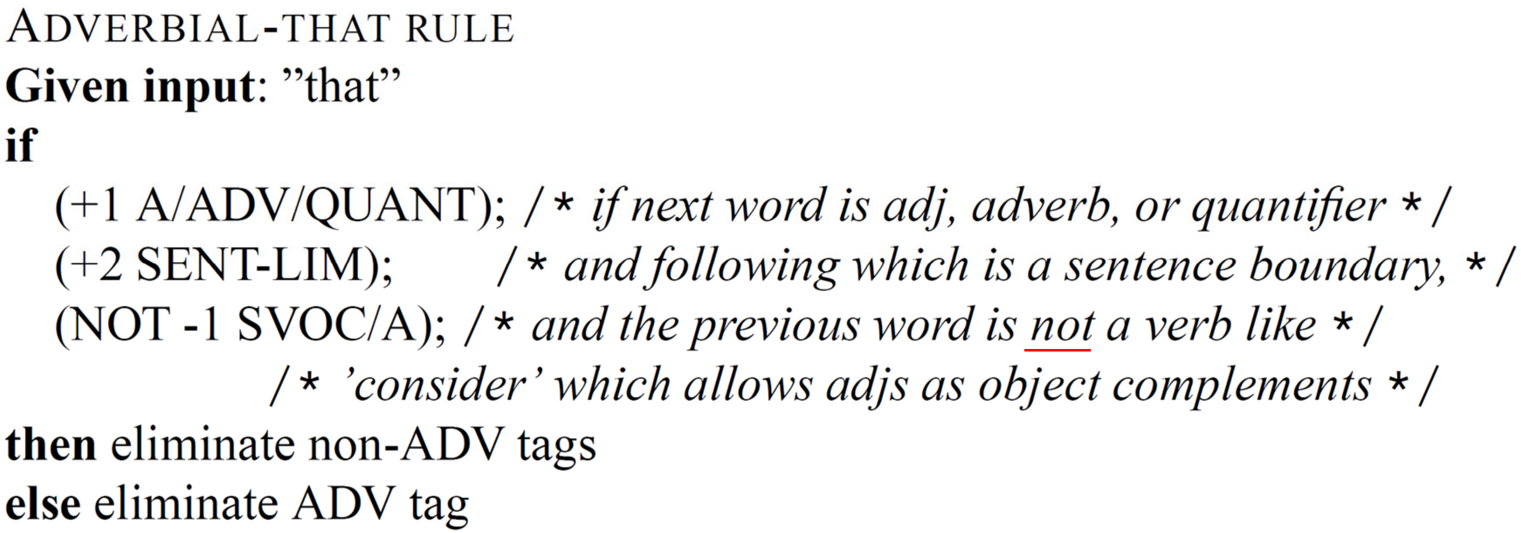
- Simple lexical entries in the ENGTWOL lexicon



| Word | POS | Additional POS features |
|---|---|---|
| smaller | ADJ | COMPARATIVE |
| entire | ADJ | ABSOLUTE ATTRIBUTIVE |
| fast | ADV | SUPERLATIVE |
| that | DET | CENTRAL DEMONSTRATIVE SG |
| all | DET | PREDETERMINER SG/PL QUANTIFIER |
| dog’s | N | GENITIVE SG |
| furniture | N | NOMINATIVE SG NOINDEFDETERMINER |
| one-third | NUM | SG |
| she | PRON | PERSONAL FEMININE NOMINATIVE SG3 |
| show | V | IMPERATIVE VFIN |
| show | V | PRESENT -SG3 VFIN |
| show | N | NOMINATIVE SG |
| shown | PCP2 | SVOO SVO SV |
| occurred | PCP2 | SV |
| occurred | V | PAST VFIN SV |

past participle

# Rule-based POS Tagging (3/3)



```
ADVERBIAL-THAT RULE
Given input: ”that”
if
   (+1 A/ADV/QUANT); /* if next word is adj, adverb, or quantifier */
   (+2 SENT-LIM);        /* and following which is a sentence boundary, */
   (NOT -1 SVOC/A); /* and the previous word is not a verb like */
                  /* ’consider’ which allows adjs as object complements */
then eliminate non-ADV tags
else eliminate ADV tag
```

Example:

It isn't **that** odd! (沒有那麼奇特的)

ADV A

I consider **that** odd. (思考那奇數)

Compliment NUM

# HMM-based Tagging (1/13)

- Also called Maximum Likelihood Tagging
  - Pick the most-likely tag for a word

- For a given sentence or words sequence , an HMM tagger chooses the tag sequence that maximizes the following probability

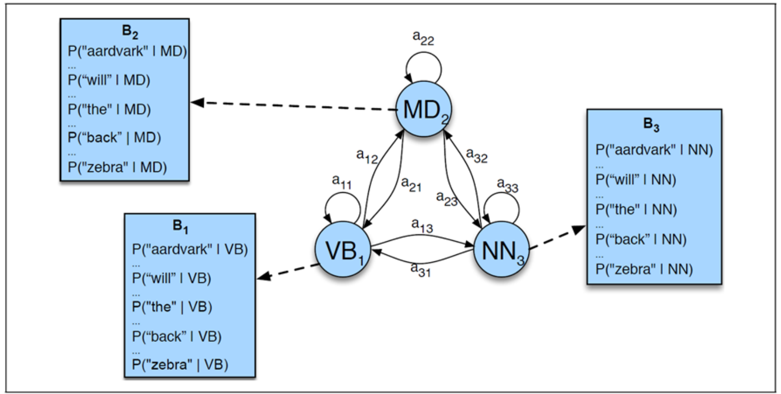


**Figure 8.9** An illustration of the two parts of an HMM representation: the *A* transition probabilities used to compute the prior probability, and the *B* observation likelihoods that are associated with each state, one likelihood for each possible observation word.

Bi-gram HMM tagger

For a word at position $n$:

$$\text{tag}_i = \underset{j}{\text{argmax}} P(\text{word}_n | \text{tag}_n = j) \cdot P(\text{tag}_n = j | \text{previous } m-1 \text{ tags})$$

word/lexical likelihood

tag sequence probability

*m*-gram HMM tagger

# HMM-based Tagging (2/13)

For a word $w_i$ at position $i$, follow Bayes$'$ rule:

$$
\begin{aligned}
t_n{}' &= \underset{t_n}{\text{argmax}} P\left(t_n \middle| w_n, t_{n-1}, t_{n-2}, \dots, t_1\right) \\
&= \underset{t_n}{\text{argmax}} \frac{P\left(w_n, t_n \middle| t_{n-1}, t_{n-2}, \dots, t_1\right)}{P\cancel{\left(w_n \middle| t_{n-1}, t_{n-2}, \dots, t_1\right)}} \\
&= \underset{t_n}{\text{argmax}} P\left(w_n, t_n \middle| t_{n-1}, t_{n-2}, \dots, t_1\right)
\end{aligned}
$$

$$
= \underset{t_n}{\text{argmax}} P\left(w_n \middle| t_n, t_{n-1}, t_{n-2}, \dots, t_1\right) P\left(t_n \middle| t_{n-1}, t_{n-2}, \dots, t_1\right)
$$

$$
\approx P\underset{t_n}{\text{argmax}}\left(w_n \middle| t_n\right) P\left(t_n \middle| \underbrace{t_{n-1}, t_{n-2}, \dots, t_{n-m+1}}_{\text{previous } m-1 \text{ tags (}m\text{-gram assumption)}}\right)
$$

However, simply picking the best tag for each word locally (in isolation)
will not result in picking the best tag sequence for an input word sequence.

# HMM-based Tagging (3/13)

- Assumptions made here
  - Words are independent of each other given their tages
    - A word's identity only depends on its tag
  - "**Limited Horizon**" and "**Time Invariant**" ("Stationary")
    - **Limited Horizon:** a word's tag only depends on the previous few tags (limited horizon) and the dependency does not change over time (time invariance)
    - **Time Invariant:** the tag dependency will not change as tag sequence appears different positions of a sentence

Cannot model long-distance relationships well!
- e.g., Wh-extraction,…

# HMM-based Tagging (4/13)

- Apply a bigram-HMM tagger to choose the best tag for a given word
  - Choose the tag $t_n$ for word $w_n$ that is most probable given the previous tag $t_{n\text{-}1}$ and current word $w_n$

$$t_n^* = \underset{j}{\operatorname{argmax}} P\left(t_n = j \middle| t_{n-1}, w_n\right)$$

  - Through some simplifying Markov assumptions

$$t_n^* = \underset{j}{\operatorname{argmax}} P(w_n | t_n = j) P\left(t_n = j \middle| t_{n-1}\right)$$

word/lexical likelihood

conditional probability of tag sequence

# HMM-based Tagging (5/13)

- Apply bigram-HMM tagger to choose the best tag for a given word

$$
\begin{aligned}
t_n^* &= \underset{j}{\operatorname{argmax}} P\left(t_n = j \middle| t_{n-1}, w_n\right) \\
&= \underset{j}{\operatorname{argmax}} \frac{P\left(t_n = j, w_n \middle| t_{n-1}\right)}{\boxed{P\left(w_n \middle| t_{n-1}\right)}} \\
&= \underset{j}{\operatorname{argmax}} P\left(t_n = j, w_n \middle| t_{n-1}\right) \\
&= \underset{j}{\operatorname{argmax}} P(w_n | t_{n-1}, t_n = j) P\left(t_n = j \middle| t_{n-1}\right) \\
&= \underset{j}{\operatorname{argmax}} P(w_n | t_n = j) P\left(t_n = j \middle| t_{n-1}\right)
\end{aligned}
$$

The same for all tags

The probability of a word only depends on its tag

# HMM-based Tagging (6/13)

- Example: Choose the best tag for a given word ("**race**" in different contexts)

1) Secretariat/NNP is /VBZ expected/VBN to/TO **race**/VB tomorrow/NN

to/TO race/???

Pretend that the previous word has already tagged

$$\overset{0.34}{P(\mathrm{VB}|\mathrm{TO})}\ \overset{0.00003}{P(\mathrm{race}|\mathrm{VB})}=0.00001$$

$$\overset{0.021}{P(\mathrm{NN}|\mathrm{TO})}\ \overset{0.00041}{P(\mathrm{race}|\mathrm{NN})}=0.000007$$

the likelihood of the noun *race* given each tag

2) People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN

# HMM-based Tagging (7/13)

- Apply bigram-HMM tagger to choose the best sequence of tags for a given sentence

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(T|W)$$

$$= \underset{T}{\operatorname{argmax}} \frac{P(T)P(W|T)}{\cancel{P(W)}}$$

$$= \underset{T}{\operatorname{argmax}} P(T)P(W|T)$$

$$= \underset{t_1,t_2,\dots,t_N}{\operatorname{argmax}} P(t_1,t_2,\dots,t_N)P(w_1,w_1,\dots,w_N|t_1,t_2,\dots,t_N)$$

$$= \underset{t_1,t_2,\dots,t_N}{\operatorname{argmax}} P(t_1)\underline{P(w_1|t_1,t_2,\dots,t_N)}\prod_{n=2}^{N}\left[P(t_n|t_1,t_2,\dots,t_{n-1})\underline{P(w_i|w_1,w_1,\dots,w_{n-1},t_1,t_2,\dots,t_N)}\right]$$

$$= \underset{t_1,t_2,\dots,t_N}{\operatorname{argmax}} P(t_1)\underline{P(w_1|t_1)}\prod_{n=2}^{N}\left[P(t_n|t_{n-m+1},t_{n-m+2},\dots,t_{n-1})\underline{P(w_n|t_n)}\right]$$

Assumptions:
- words are independent of each other
- a word's identity only depends on its tag

The probability of a word only depends on its tag

Tag M-gram assumption

# HMM-based Tagging (8/13)

- The Viterbi algorithm for the bigram-HMM tagger

Andrew Viterbi

https://en.wikipedia.org/wiki/Andrew_Viterbi
(the co-founder of Qualcomm Inc. with Jacobs)

1. Initialization $\delta_1(j) = \pi_j P(w_1|t_1 = j), 1 \le j \le J \quad , \pi_j = P(t_1 = j)$

2. Induction $\delta_n(j) = \left[\max_{1 \le i \le J} \delta_{n-1}(i) P(t_n = j | t_{n-1} = i)\right] P(w_n|t_n = j), \qquad 2 \le n \le \underline{N} \quad 1 \le i,j \le \underline{J}$

$$\psi_n(j) = \underset{1 \le i \le J}{\operatorname{argmax}} \left[\delta_{n-1}(i) P(t_n = j | t_{n-1} = i)\right]$$

The length of the input word sequence

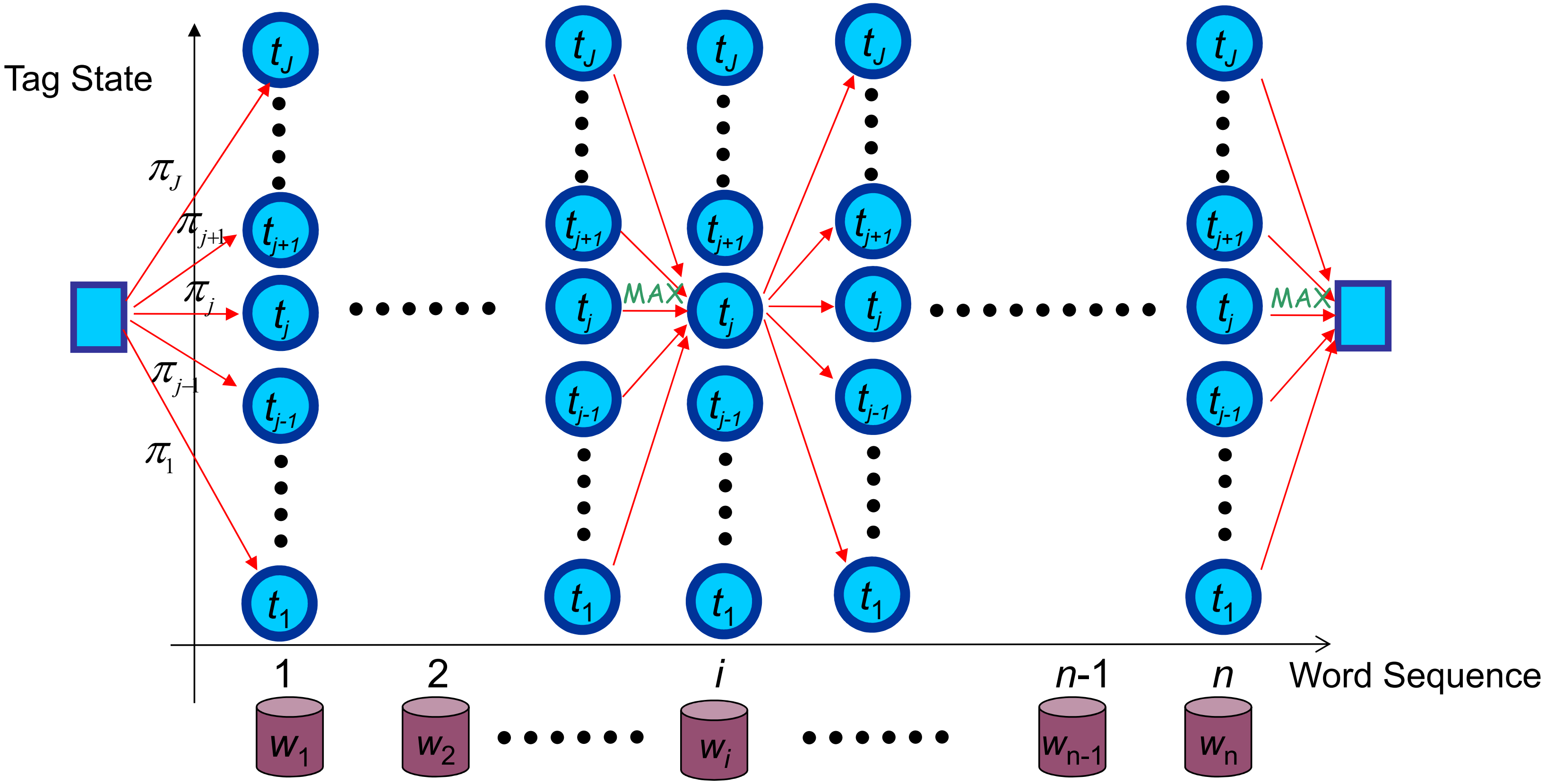
The total indices of tags

3. Termination $X_N = \underset{1 \le j \le J}{\operatorname{argmax}} \delta_n(j)$

```
for n := $N - 1$ to 1 step $-1$ do
    $X_n = \psi_n(X_{n+1})$
end
```

# HMM-based Tagging (9/13)

- The Viterbi algorithm for the bigram-HMM tagger
  - States: distinct tags
  - Observations: input word generated by each state



# HMM-based Tagging (10/13)

- Apply trigram-HMM tagger to choose the best sequence of tags for a given sentence
  - **When trigram model is used**

$$\hat{T} = \underset{t_1, t_2, \ldots, t_N}{\operatorname{argmax}} \left[ P(t_1) P(t_2 | t_1) \prod_{n=3}^{N} P(t_n | t_{n-2}, t_{n-1}) \right] \left[ \prod_{n=1}^{N} P(w_n | t_n) \right]$$

  - Maximum likelihood estimation based on the relative frequencies observed in the pre-tagged training corpus (labeled data)

$$P_{ML}(t_i | t_{i-2}, t_{i-1}) = \frac{c(t_{i-2} t_{i-1} t_i)}{\sum_j c(t_{i-2} t_{i-1} t_j)}$$

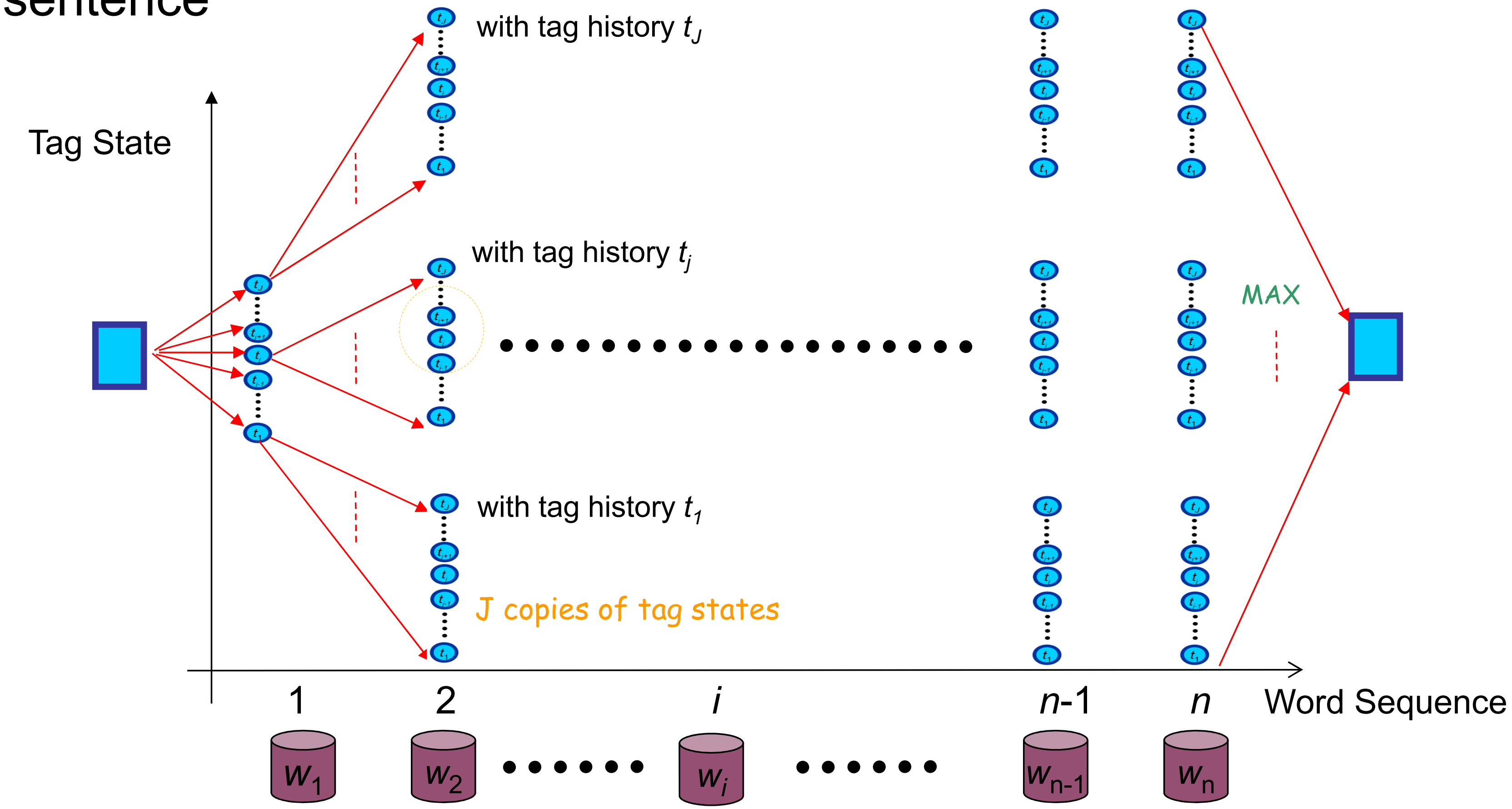
$$P_{ML}(w_i | t_i) = \frac{c(w_i, t_i)}{\sum_j c(w_j, t_i)}$$

Smoothing or linear interpolation are needed !

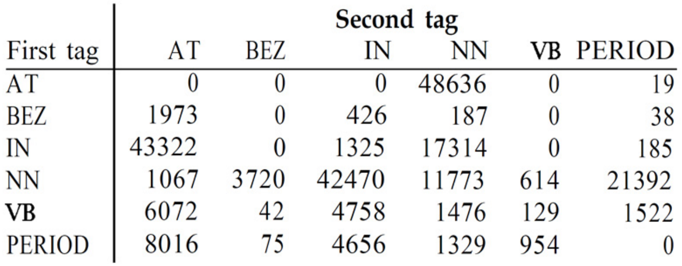
$$P_{smoothed}(t_i | t_{i-2}, t_{i-1})$$
$$= \alpha \cdot P_{ML}(t_i | t_{i-2}, t_{i-1}) + \beta \cdot P_{ML}(t_i | t_{i-1}) + (1 - \alpha - \beta) \cdot P_{ML}(t_i)$$

# HMM-based Tagging (11/13)

- Apply trigram-HMM tagger to choose the best sequence of tags for a given sentence



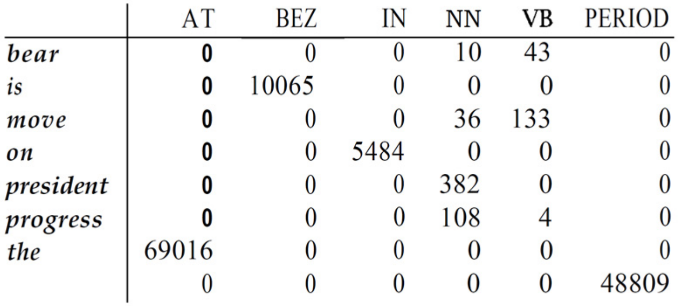
# HMM-based Tagging (12/13)



| First tag | Second tag<br>AT | BEZ | IN | NN | **VB** | PERIOD |
|---|---|---|---|---|---|---|
| AT | 0 | 0 | 0 | 48636 | 0 | 19 |
| BEZ | 1973 | 0 | 426 | 187 | 0 | 38 |
| IN | 43322 | 0 | 1325 | 17314 | 0 | 185 |
| NN | 1067 | 3720 | 42470 | 11773 | 614 | 21392 |
| **VB** | 6072 | 42 | 4758 | 1476 | 129 | 1522 |
| PERIOD | 8016 | 75 | 4656 | 1329 | 954 | 0 |

Table 10.3 Idealized counts of some tag transitions in the Brown Corpus. For example, NN occurs 48636 times after AT.

$$P(t_j|t_i) = \frac{c(t_i t_j)}{\sum_{j'} c(t_i t_{j'})}$$



| | AT | BEZ | IN | NN | **VB** | PERIOD |
|---|---|---|---|---|---|---|
| *bear* | **0** | 0 | 0 | 10 | 43 | 0 |
| *is* | **0** | 10065 | 0 | 0 | 0 | 0 |
| *move* | **0** | 0 | 0 | 36 | 133 | 0 |
| *on* | **0** | 0 | 5484 | 0 | 0 | 0 |
| *president* | **0** | 0 | 0 | 382 | 0 | 0 |
| *progress* | **0** | 0 | 0 | 108 | 4 | 0 |
| *the* | 69016 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 48809 |

Table 10.4 Idealized counts for the tags that some words occur with in the Brown Corpus. For example, 36 occurrences of ***move*** are with the tag NN.

$$P\left(w_n \middle| t_j\right) = \frac{c(w_n, t_j)}{\sum_{n'} c(w_{n'}, t_j)}$$

- Probability smoothing of $P(t_j|t_i)$ and $P\left(w_n \middle| t_j\right)$ is necessary

# HMM-based Tagging (13/13)

$$P\left(w_n \middle| t_j\right) = \frac{c\left(w_n, t_j\right)}{\sum_{n'} c\left(w_{n'}, t_j\right)}$$

$$P\left(t_i \middle| t_j\right) = \frac{c\left(t_j t_i\right)}{\sum_i c\left(t_j t_j\right)}$$

- Probability re-estimation based on unlabeled data
  - **EM** (Expectation-Maximization) algorithm is applied
    - Start with a dictionary that lists which tags can be assigned to which words
      - $P\left(w_n \middle| t_j\right)$: word likelihood functions (emission probabilities) can be estimated
      - $P\left(t_j \middle| t_i\right)$: tag transition probabilities set to be equal
    - **EM** algorithm learns (re-estimates) the word likelihood function for each tag and the tag transition probabilities

  - However, a tagger trained on hand-tagged data worked better than one trained via **EM**
    - Treat the model as a Markov Model in training but treat them as a Hidden Markov Model in tagging

Secretariat/NNP is /VBZ expected/VBN to/TO race/VB tomorrow/NN

# Transformation-based Tagging (1/8)

- Also called Brill tagging (proposed by Eric Brill, 1995)
  - An instance of Transformation-Based Learning (TBL)
  - Draw inspiration from both the rule-based and stochastic taggers
- Notion
  - Like the **rule-based approach**, TBL is based on rules that specify what tags should be assigned to what word
  - Like the **stochastic approach**, rules are automatically induced from the data by the machine learning (ML) technique
- Note that TBL is a supervised learning technique
  - It assumes a pre-tagged training corpus

# Transformation-based Tagging (2/8)

- How the TBL rules are learned
  - **Three major stages**
    1. Label every word with its most-likely tag using a set of tagging rules (use the broadest rules at first)
    2. Examine every possible transformation (to rewrite rules), and select the one that results in the most improved tagging (supervised! should compare to the pre-tagged corpus )
    3. Re-tag the data according this rule
  - The above three stages are repeated until some stopping criterion is reached
    - Such as insufficient improvement over the previous pass
  - An ordered list of transformations (rules) can be finally obtained

# Transformation-based Tagging (3/8)

- Example

1

*P*(NN|race)=0.98<br>
*P*(VB|race)=0.02

So, race will be initially coded as NN<br>
(label every word with its most-likely tag)

2

(a). is/VBZ expected/VBN to/To race/NN tomorrow/NN

(b). the/DT race/NN for/IN outer/JJ space/NN

Refer to the correct tag Information of each word, and find the tag of *race* in (a) is wrong

3

Learn/pick a most suitable transformation rule: (by examining every possible transformation)

**Change NN to VB while the previous tag is TO**

Rewrite rule: expected/VBN to/To race/NN → expected/VBN to/To race/VB

These three stages are repeated until some stopping criterion is reached, such as insufficient improvement over the previous pass.

# Transformation-based Tagging (4/8)

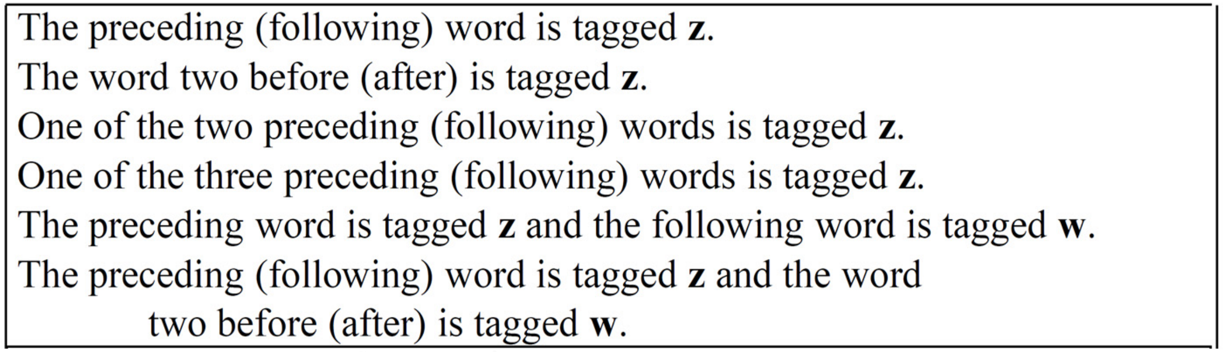
- Templates (abstracted transformations)
  - The set of possible transformations may be **infinite**
  - Should **limit** the set of transformations
  - The design of a **small set of templates** (abstracted transformations) is needed

E.g., a strange rule like:

**transform NN to VB if the previous word was "IBM" and the word "the" occurs between 17 and 158 words before that**

# Transformation-based Tagging (5/8)

- Possible templates (abstracted transformations)



The preceding (following) word is tagged **z**.
The word two before (after) is tagged **z**.
One of the two preceding (following) words is tagged **z**.
One of the three preceding (following) words is tagged **z**.
The preceding word is tagged **z** and the following word is tagged **w**.
The preceding (following) word is tagged **z** and the word
two before (after) is tagged **w**.

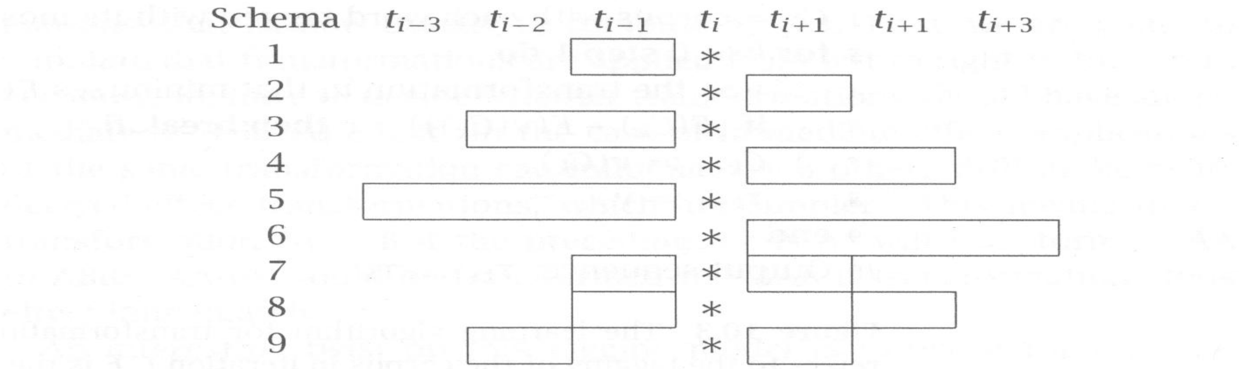


**Table 10.7** Triggering environments in Brill's transformation-based tagger. Examples: Line 5 refers to the triggering environment "Tag $t^j$ occurs in one of the three previous positions"; Line 9 refers to the triggering environment "Tag $t^j$ occurs two positions earlier and tag $t^k$ occurs in the following position."

Brill's (1995) templates.

Each begins with *"Change tag* **a** *to tag* **b** *when:… "*.

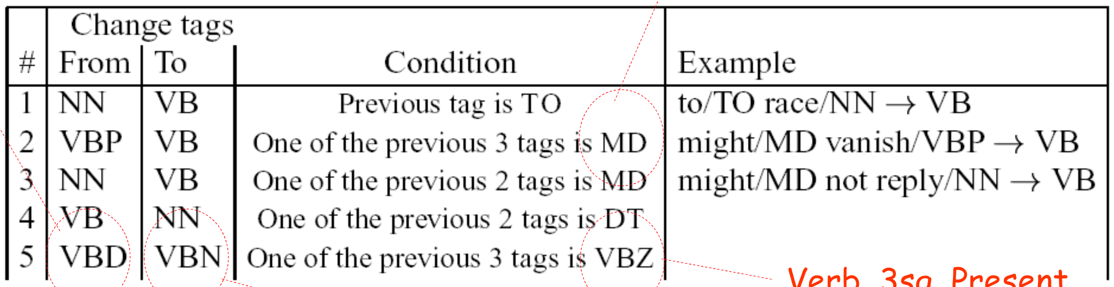
The variables **a**, **b**, **z**, and **w** range over parts of speech.

# Transformation-based Tagging (6/8)

- Learned transformations

Verb, 3sg, past tense

Modal verbs (should, can,…)



|  | Change tags |  |  |  |
|---|---|---|---|---|
| # | From | To | Condition | Example |
| 1 | NN | VB | Previous tag is TO | to/TO race/NN → VB |
| 2 | VBP | VB | One of the previous 3 tags is MD | might/MD vanish/VBP → VB |
| 3 | NN | VB | One of the previous 2 tags is MD | might/MD not reply/NN → VB |
| 4 | VB | NN | One of the previous 2 tags is DT |  |
| 5 | VBD | VBN | One of the previous 3 tags is VBZ |  |

Verb, past participle

Verb, 3sg, Present

Rules learned by Brill's original tagger

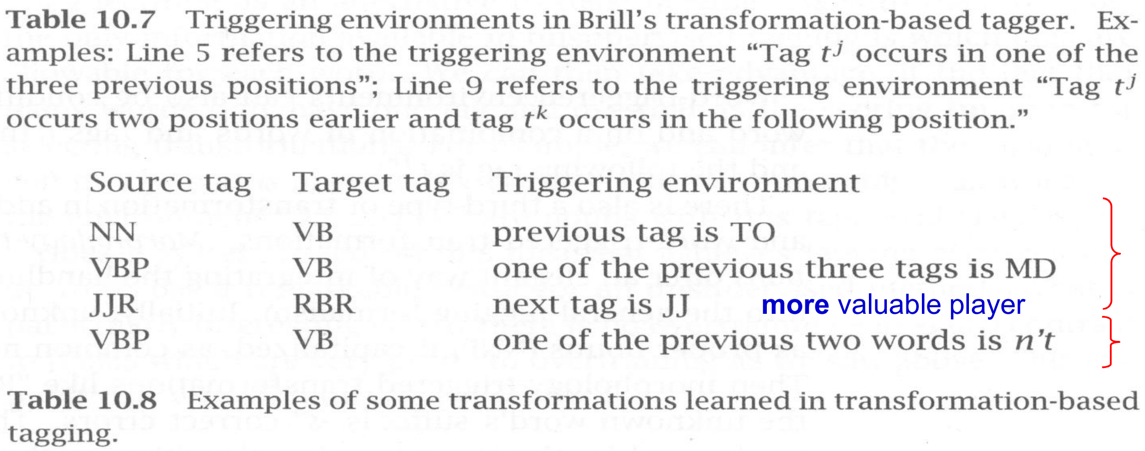


**Table 10.7** Triggering environments in Brill's transformation-based tagger. Examples: Line 5 refers to the triggering environment "Tag $t^j$ occurs in one of the three previous positions"; Line 9 refers to the triggering environment "Tag $t^j$ occurs two positions earlier and tag $t^k$ occurs in the following position."

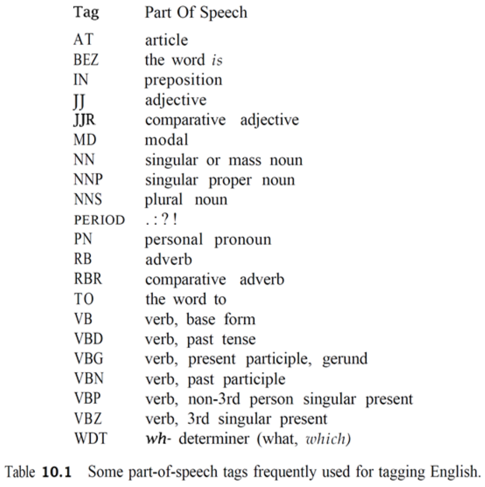
| Source tag | Target tag | Triggering environment |
|---|---|---|
| NN | VB | previous tag is TO |
| VBP | VB | one of the previous three tags is MD |
| JJR | RBR | next tag is JJ |
| VBP | VB | one of the previous two words is *n't* |

**more** valuable player

Constraints for tags

Constraints for words

**Table 10.8** Examples of some transformations learned in transformation-based tagging.

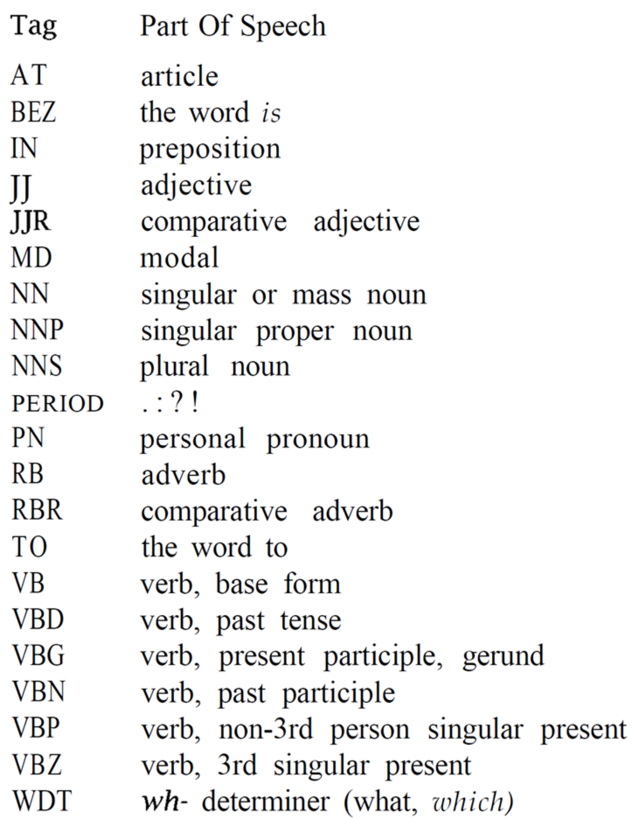


| Tag | Part Of Speech |
|---|---|
| AT | article |
| BEZ | the word *is* |
| IN | preposition |
| JJ | adjective |
| JJR | comparative adjective |
| MD | modal |
| NN | singular or mass noun |
| NNP | singular proper noun |
| NNS | plural noun |
| PERIOD | . : ? ! |
| PN | personal pronoun |
| RB | adverb |
| RBR | comparative adverb |
| TO | the word to |
| VB | verb, base form |
| VBD | verb, past tense |
| VBG | verb, present participle, gerund |
| VBN | verb, past participle |
| VBP | verb, non-3rd person singular present |
| VBZ | verb, 3rd singular present |
| WDT | *wh-* determiner (what, *which*) |

Table **10.1** Some part-of-speech tags frequently used for tagging English.

# Transformation-based Tagging (7/8)

- Reference for tags used in the previous slide

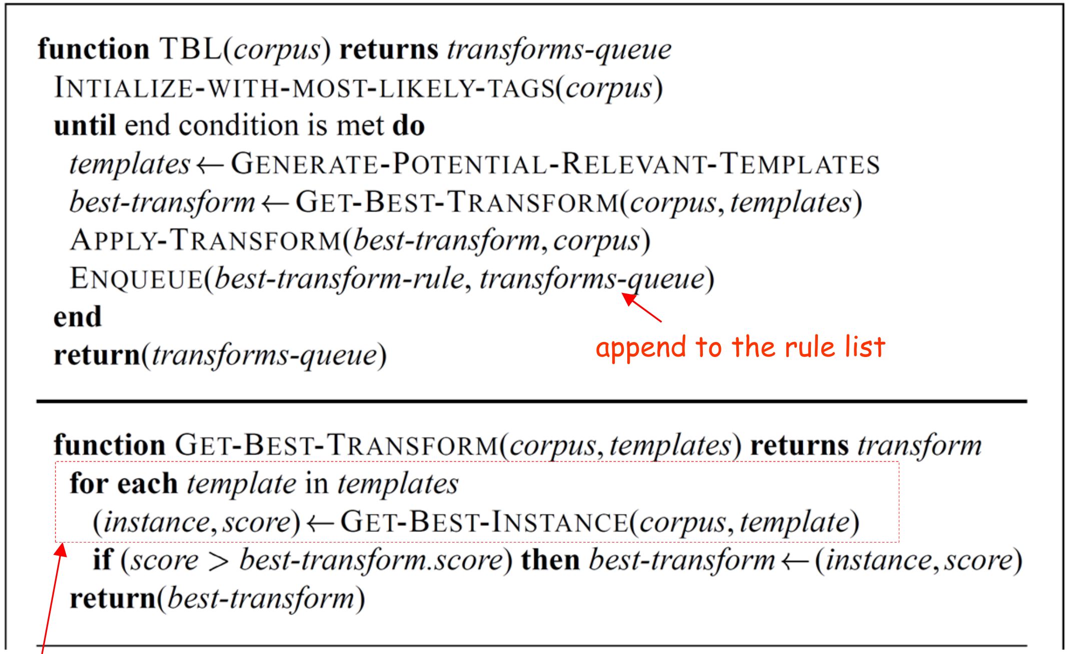


| Tag | Part Of Speech |
|---|---|
| AT | article |
| BEZ | the word *is* |
| IN | preposition |
| JJ | adjective |
| JJR | comparative adjective |
| MD | modal |
| NN | singular or mass noun |
| NNP | singular proper noun |
| NNS | plural noun |
| PERIOD | . : ? ! |
| PN | personal pronoun |
| RB | adverb |
| RBR | comparative adverb |
| TO | the word to |
| VB | verb, base form |
| VBD | verb, past tense |
| VBG | verb, present participle, gerund |
| VBN | verb, past participle |
| VBP | verb, non-3rd person singular present |
| VBZ | verb, 3rd singular present |
| WDT | *wh-* determiner (what, *which*) |

Table **10.1** Some part-of-speech tags frequently used for tagging English.

# Transformation-based Tagging (8/8)

- Algorithm

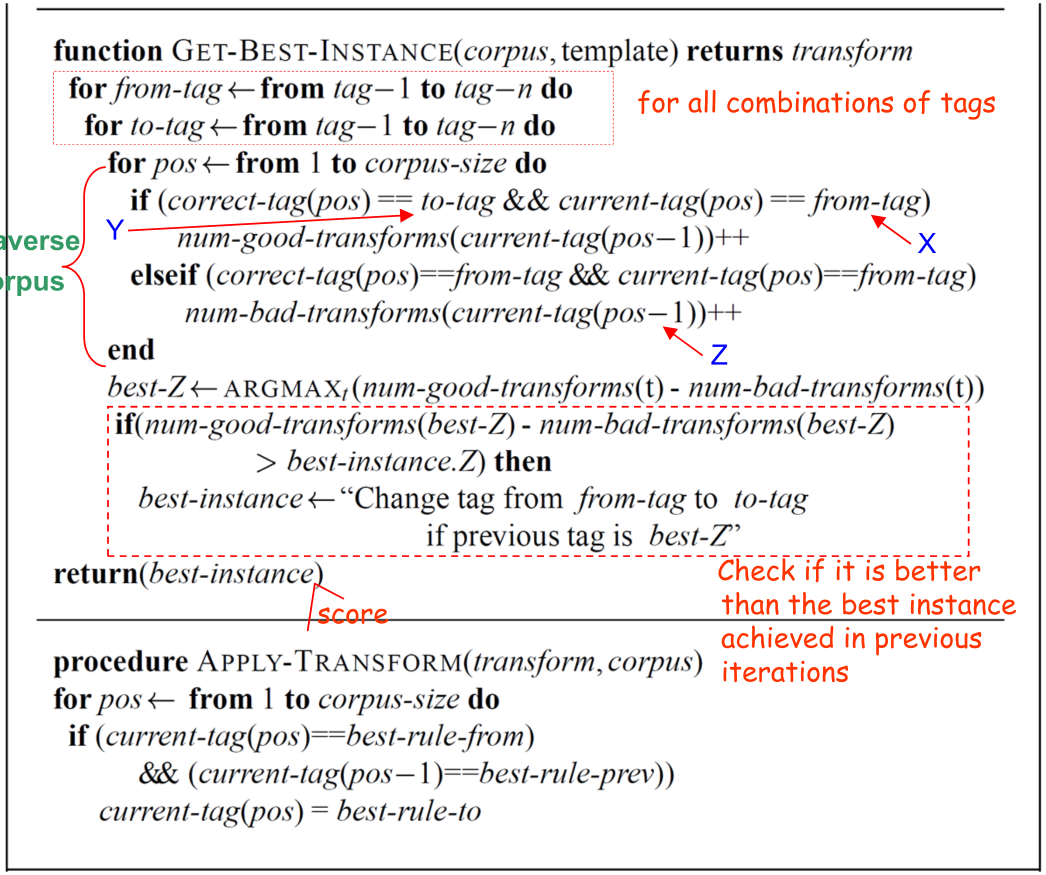


```
function TBL(corpus) returns transforms-queue
  INTIALIZE-WITH-MOST-LIKELY-TAGS(corpus)
  until end condition is met do
    templates ← GENERATE-POTENTIAL-RELEVANT-TEMPLATES
    best-transform ← GET-BEST-TRANSFORM(corpus, templates)
    APPLY-TRANSFORM(best-transform, corpus)
    ENQUEUE(best-transform-rule, transforms-queue)
  end
  return(transforms-queue)

function GET-BEST-TRANSFORM(corpus, templates) returns transform
  for each template in templates
    (instance, score) ← GET-BEST-INSTANCE(corpus, template)
    if (score > best-transform.score) then best-transform ← (instance, score)
  return(best-transform)
```

append to the rule list

Get best instance
for each transformation



```
function GET-BEST-INSTANCE(corpus, template) returns transform
  for from-tag ← from tag−1 to tag−n do
    for to-tag ← from tag−1 to tag−n do
      for pos ← from 1 to corpus-size do
        if (correct-tag(pos) == to-tag && current-tag(pos) == from-tag)
            num-good-transforms(current-tag(pos−1))++
        elseif (correct-tag(pos)==from-tag && current-tag(pos)==from-tag)
            num-bad-transforms(current-tag(pos−1))++
      end
      best-Z ← ARGMAX_t(num-good-transforms(t) - num-bad-transforms(t))
      if(num-good-transforms(best-Z) - num-bad-transforms(best-Z)
                > best-instance.Z) then
        best-instance ← "Change tag from from-tag to to-tag
                                if previous tag is best-Z"
  return(best-instance)

procedure APPLY-TRANSFORM(transform, corpus)
  for pos ← from 1 to corpus-size do
    if (current-tag(pos)==best-rule-from)
        && (current-tag(pos−1)==best-rule-prev))
      current-tag(pos) = best-rule-to
```

for all combinations of tags

traverse corpus

Y

X

Z

score

Check if it is better
than the best instance
achieved in previous
iterations

The **GET_BEST_INSTANCE** procedure in the example algorithm is "Change tag from X to Y if the previous tag is Z".

# CRF-based Tagging (1/2)

- Conditional Random Field (CRF) is a model-based approach
  - A discriminative sequence model based on log-linear models
  - **Linear chain CRF**: a version of the CRF that conditions its computation only on previous transitions that is most commonly used for NLP (with sequential structures)

CRF computes $P(T|W)$ directly:

**global feature**

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(T|W) = \underset{T}{\operatorname{argmax}} \frac{\exp(\sum_{k=1}^{K} \alpha_k F_k(W,T))}{\sum_{T'} \exp(\sum_{k=1}^{K} \alpha_k F_k(W,T'))}$$

these $K$ functions $F_k(W,T)$ global features

Recall that, HMM computes $P(Y|X)$ indirectly with Bayes' rule:

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} P(T)P(W|T) \\ &= \underset{t_1,t_2,\ldots,t_N}{\operatorname{argmax}} P(t_1)P(w_1|t_1) \prod_{n=2}^{N} P(t_n|t_{n-1})P(w_n|t_n)\end{aligned}$$

# CRF-based Tagging (2/2)

- The formulation of CRF can be simplified to

$$\hat{T} = \underset{T}{\operatorname{argmax}} P(T|W) = \frac{1}{Z(W)} \exp(\textstyle\sum_{k=1}^{K} \alpha_k \, F_k(W,T))$$

$$\text{where} \quad Z(W) = \textstyle\sum_{T'} \exp(\sum_{k=1}^{K} \alpha_k \, F_k(W,T'))$$

  - We then compute $F_k(W,T)$ by decomposing it into a sum of local features for each position $n$ in $T$

$$F_k(W,T) = \sum_{n=1}^{N} f_n(\underline{t_n, t_{n-1}}, W, n)$$

linear-chain properties

Some legal features $f_n(t_n, t_{n-1}, W, n)$ with values 0 or 1

$$\mathrm{I}\{w_n = the, t_n = \mathrm{DET}\}$$
$$\mathrm{I}\{t_n = \mathrm{PROPN}, w_{n+1} = Street, y_{n-1} = \mathrm{NUM}\}$$
$$\mathrm{I}\{t_n = \mathrm{VERB}, y_{n-1} = \mathrm{AUX}\}$$

  - Each of these **local features** $f_n$ of a **linear-chain CRF** is allowed to make use of the current output token $t_n$ , the previous output token $t_{n-1}$, the entire input string $W$ (or any subpart of it), and the current position $n$

# Inference and Training for CRF (1/2)

- How do we find the best tag sequence $\hat{T}$ for a given input $W$?

$$
\begin{aligned}
\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\
&= \underset{T}{\operatorname{argmax}} \frac{1}{Z(W)} \exp\left(\sum_{k=1}^{K} \alpha_k \underline{F_k(W,T)}\right) \\
&= \underset{T}{\operatorname{argmax}} \exp\left(\sum_{k=1}^{K} \alpha_k \underline{\sum_{n=1}^{N} f_n(t_n, t_{n-1}, W, n)}\right) \\
&= \underset{T}{\operatorname{argmax}} \exp\left(\sum_{i=1}^{N} \sum_{k=1}^{K} \alpha_k f_n(t_n, t_{n-1}, W, n)\right)
\end{aligned}
$$

$$\boxed{F_k(W,T) = \sum_{n=1}^{N} f_n(t_n, t_{n-1}, W, n)}$$

- Just as with HMM, we can resort to **the Viterbi algorithm**
  - Because, like HMM, linear-chain CRF depends at each timestep on only one previous output token $t_{n-1}$

# Inference and Training for CRF (2/2)

- We can make inference with CRF in an autoregressive manner

$$v_n(j) = \max_{i=1}^{J} v_{n-1}(i) \sum_{k=1}^{K} \alpha_k f_n(t_n = j, t_{n-1} = i, W, n) \quad \underline{1} \le i, j \le J, 1 \le n \le \underline{N}$$

The total indices of tags

The length of the input word sequence

- In the training phase, given a sequence of observations, feature functions, and corresponding outputs, we use stochastic gradient descent to train the weights $\alpha_k$ to maximize the log-likelihood of the training corpus

# Exemplar Features of a Liner-Chain CRF

- Some legal features representing common situations might be the following

$$\mathbb{1}\{x_i = \textit{the},\ y_i = \text{DET}\}$$
$$\mathbb{1}\{y_i = \text{PROPN},\ x_{i+1} = \textit{Street},\ y_{i-1} = \text{NUM}\}$$
$$\mathbb{1}\{y_i = \text{VERB},\ y_{i-1} = \text{AUX}\}$$

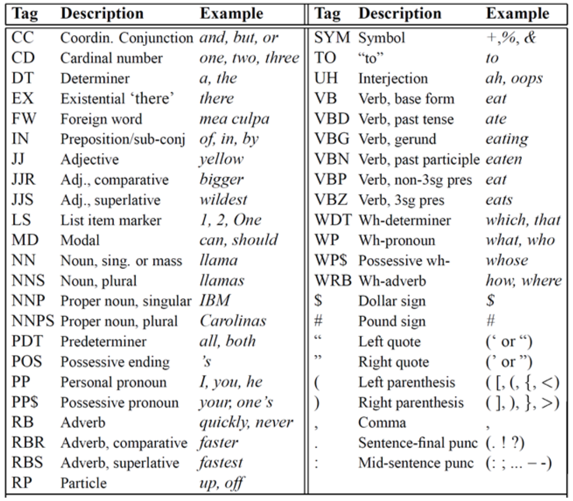
- Specific features are automatically populated by using feature templates

$$\langle y_i, x_i \rangle, \langle y_i, y_{i-1} \rangle, \langle y_i, x_{i-1}, x_{i+2} \rangle$$

$$f_{3743}\text{: } y_i = \text{VB and } x_i = \text{back}$$
$$f_{156}\text{: } y_i = \text{VB and } y_{i-1} = \text{MD}$$
$$f_{99732}\text{: } y_i = \text{VB and } x_{i-1} = \text{will and } x_{i+2} = \text{bill}$$

# Multiple Tags and Multi-part Words

- Multiple tags
  - A word is ambiguous between multiple tags and it is impossible or very difficult to disambiguate, so multiple tags is allowed, e.g.
    - adjective versus preterite versus past participle (JJ/VBD/VBN)
    - adjective versus noun as prenominal modifier (JJ/NN)



| Tag | Description | Example | Tag | Description | Example |
|---|---|---|---|---|---|
| CC | Coordin. Conjunction | *and, but, or* | SYM | Symbol | +,%, & |
| CD | Cardinal number | *one, two, three* | 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, singular | *IBM* | $ | Dollar sign | *$* |
| NNPS | Proper noun, plural | *Carolinas* | # | Pound sign | *#* |
| PDT | Predeterminer | *all, both* | " | Left quote | (' or ") |
| POS | Possessive ending | *'s* | " | Right quote | (' or ") |
| PP | Personal pronoun | *I, you, he* | ( | Left parenthesis | ( [, (, {, <) |
| PP$ | 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* | | | |

(Penn Treebank Tagset)

- Multi-part words
  - Certain words are split; or, some adjacent words are treated as a single word

(Penn Treebank Tagset) would/MD n't/RB Children/NNS 's/POS

treated as separate words by splitting contractions and the 's-genitive from their stems

(C7 tagset) in terms of (in/II31 terms/II32 of/II33)

II: general preposition

treated as a single word by adding numbers to each tag

# Tagging of Unknown Words (1/4)

- Unknown words are a major problem for taggers
    - Different accuracy of taggers over different corpora is often determined by the proportion of unknown words

- How to guess the part of speech of unknown words?

  1) Simplest unknown-word algorithm

  2) Slightly more complex algorithm

  3) More powerful unknown-word algorithm

# Tagging of Unknown Words (2/4)

## 1) Simplest unknown-word algorithm

- Pretend that each unknown word is ambiguous among all possible tags, with equal probability ($P(UNK|t_1)$=$P(UNK|t_2) = \cdots$)
  - Lose/ignore lexical information for unknown words
- Must rely solely on the contextual POS-trigram (syntagmatic information) to suggest the proper tag

$$\hat{T} = \underset{t_1,t_2,..,t_N}{\operatorname{argmax}} \left[ P(t_1)P(t_2|t_1) \prod_{n=3}^{N} P(t_n|t_{n-2},t_{n-1}) \right] \left[ \prod_{n=1}^{N} P(w_n|t_n) \right]$$

what if $w_n$ is $UNK$

## 2) Slightly more complex algorithm

- Based on the idea that the probability distribution of tags over unknown words is very similar to the distribution of tags over words that occurred only once (singletons) in a training set
- The likelihood for an unknown word is determined by the average of the distribution over all singleton in the training set (similar to Good-Turing? )

Nouns or Verbs $P(w_n|t_n)$?

# Tagging of Unknown Words (3/4)

## 3) One more powerful unknown-word algorithm

- Hand-designed features
  - The information about how the word is spelled (inflectional and derivational features), e.g.:
    - Words end with s (→plural nouns)
    - Words end with ed (→past participles)
  - The information of word capitalization (initial or non-initial) and hyphenation

$$P(w_n|t_n) = p(\text{unknown}-\text{word}|t_n) \cdot p(\text{captial}|t_n) \cdot p(\text{endings/hyph}|t_n)$$

what if $w_n$ is $UNK$

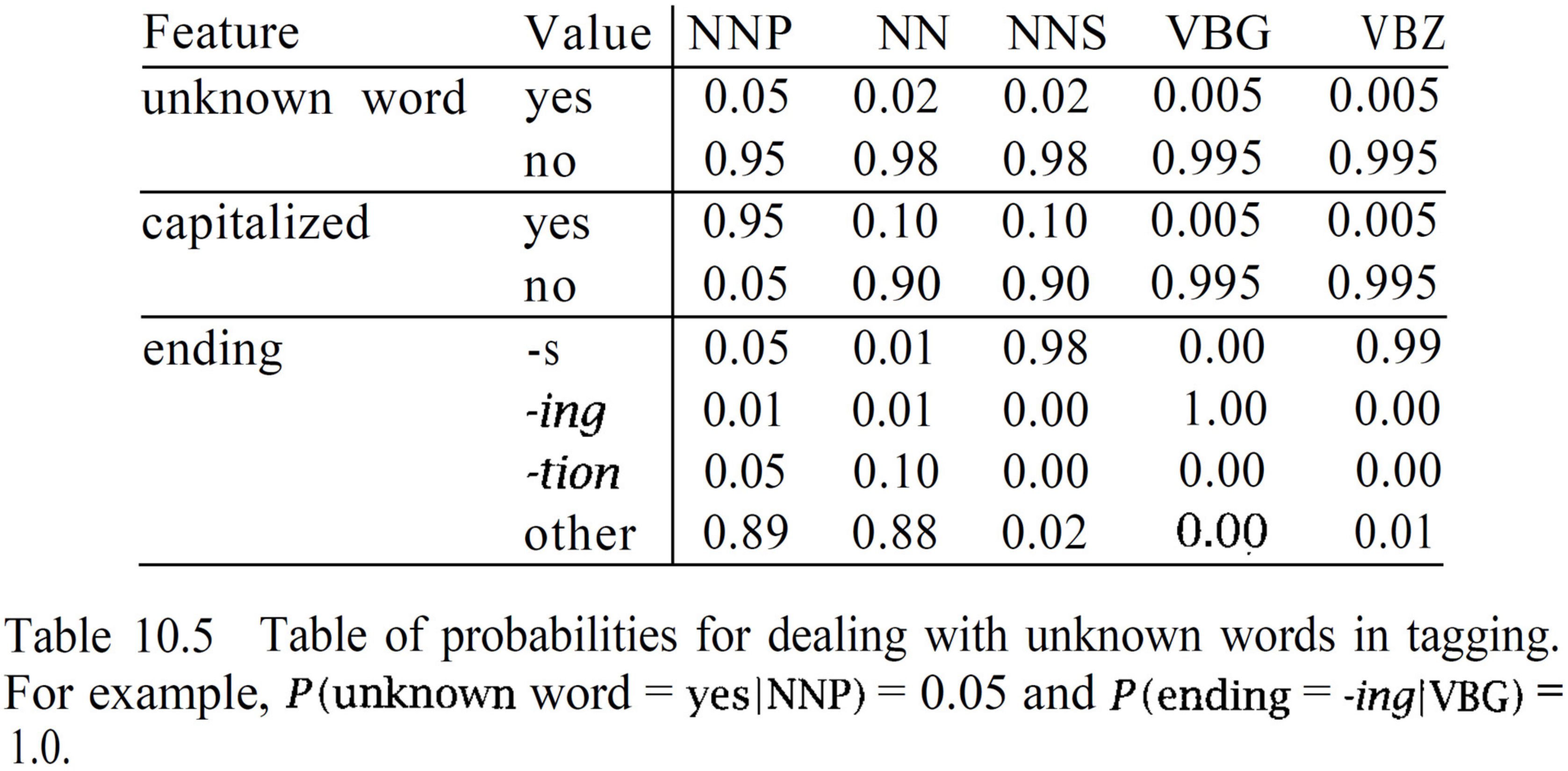
- Features induced by machine learning

  Assumption: independence between features

  - E.g.: TBL algorithm uses templates to induce useful English inflectional and derivational features and hyphenation

The first N letters of the word
The last N letters of the word

# Tagging of Unknown Words (4/4)



| Feature | Value | NNP | NN | NNS | VBG | VBZ |
|---|---|---|---|---|---|---|
| unknown word | yes | 0.05 | 0.02 | 0.02 | 0.005 | 0.005 |
| | no | 0.95 | 0.98 | 0.98 | 0.995 | 0.995 |
| capitalized | yes | 0.95 | 0.10 | 0.10 | 0.005 | 0.005 |
| | no | 0.05 | 0.90 | 0.90 | 0.995 | 0.995 |
| ending | -s | 0.05 | 0.01 | 0.98 | 0.00 | 0.99 |
| | *-ing* | 0.01 | 0.01 | 0.00 | 1.00 | 0.00 |
| | *-tion* | 0.05 | 0.10 | 0.00 | 0.00 | 0.00 |
| | other | 0.89 | 0.88 | 0.02 | 0.00 | 0.01 |

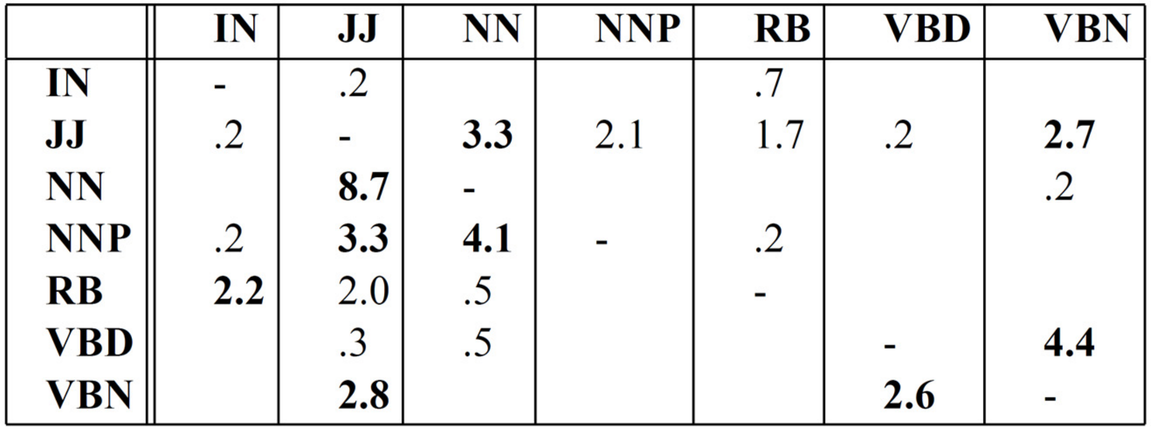
Table 10.5 Table of probabilities for dealing with unknown words in tagging. For example, $P(\text{unknown word} = \text{yes}|\text{NNP}) = 0.05$ and $P(\text{ending} = \textit{-ing}|\text{VBG}) = 1.0$.

# Evaluation of Taggers

- Compare the tagged results with a human labeled ***Gold Standard*** test set in percentages of correction
  - Most tagging algorithms have an accuracy of around 96~97% for the sample tagsets like the Penn Treebank set
  - Upper bound (human ceiling) and lower bound (baseline)
    - *Ceiling*: is achieved by seeing how well humans do on the task
      - A 3~4% margin of error
    - *Baseline*: is achieved by using the **unigram** most-like tags for each word
      - 90~91% accuracy can be attained

# Error Analysis

- Confusion matrix



| | IN | JJ | NN | NNP | RB | VBD | VBN |
|---|---|---|---|---|---|---|---|
| **IN** | - | .2 | | | .7 | | |
| **JJ** | .2 | - | **3.3** | 2.1 | 1.7 | .2 | **2.7** |
| **NN** | | **8.7** | - | | | | .2 |
| **NNP** | .2 | **3.3** | **4.1** | - | .2 | | |
| **RB** | **2.2** | 2.0 | .5 | | - | | |
| **VBD** | | .3 | .5 | | | - | **4.4** |
| **VBN** | | **2.8** | | | | **2.6** | - |

(%)

- Major problems facing current taggers
  - NN (noun) versus NNP (proper noun) and JJ (adjective)
  - RP (particle) versus RB (adverb) versus JJ
  - VBD (past tense verb) versus VBN (past participle verb) versus JJ

# Applications of POS Tagging (1/3)

- Tell what words are likely to occur in a word's vicinity
  - E.g. the vicinity of the possessive or person pronouns
- Tell the pronunciation of a word
  - DIScount (noun) and disCOUNT (verb) …
- Advanced ASR language models
  - Word-class N-grams
- Partial parsing
  - A simplest one: find the noun phrases (names) or other phrases in a sentence

# Applications of POS Tagging (2/3)

- Information retrieval
  - Word stemming
  - Help select out nouns or important words from a doc
  - Phrase-level information

    United, States, of, America → "United States of America"
    secondary, education → "secondary education"

    - Phrase normalization

    Book publishing, publishing of books

- Information extraction
  - Semantic tags or categories

# Applications of POS Tagging (3/3)

- Question Answering
  - Answer a user query that is formulated in the form of a question by return an appropriate noun phrase such as a location, a person, or a date
    - E.g. "Who killed President Kennedy?"

In summary, the role of taggers appears to be a fast lightweight component that gives sufficient information for many applications

- But not always a desirable preprocessing stage for all applications
- Many probabilistic parsers are now good enough!

# Class-based *N*-grams

- Use the lexical tag/category/class information to augment the *N*-gram models

$$P\left(w_n \middle| w_{n-N+1}^{n-1}\right) = P(w_n|c_n)P\left(c_n \middle| c_{n-N+1}^{n-1}\right)$$

probability of a word given the tag

probability of a tag given the previous *N*-1 tags

  - Maximum likelihood estimation

$$P\left(w_i \middle| c_j\right) = \frac{C(w)}{C(c)}$$

$$P(c_j|c_k) = \frac{C\left(c_k c_j\right)}{\sum_l C\left(c_l c_j\right)}$$

Constraints: a word may only belong to one lexical category

# Epilogue

- Words in a language
  - A relatively small set of **closed class** words, which are often highly frequent, generally act as **function words**, and can be very ambiguous in their POS tags
  - **Open class** words generally include various kinds of **nouns**, **verbs**, **adjectives**
  - There are a number of part-of-speech coding schemes, based on **tagsets** of between 40 and 200 tags
- Part-of-speech tagging is the process of assigning a part-of-speech (POS) label to each of a sequence of words
  - Taggers can be characterized as **rule-based, stochastic** or their hybrids
- Taggers are often evaluated by comparing their output from a test set to human labels for that test set

# Named-Entity Recognition (1/4)

- Named entities (NE) include
  - Proper nouns as names for persons, locations, organizations, artifacts and so on
  - Temporal expressions such as “Oct. 10 2003” or “1:40 p.m.”
  - Numerical quantities such as “fifty dollars” or “thirty percent”

- Temporal expressions and numerical quantities can be easily modeled and extracted by rules

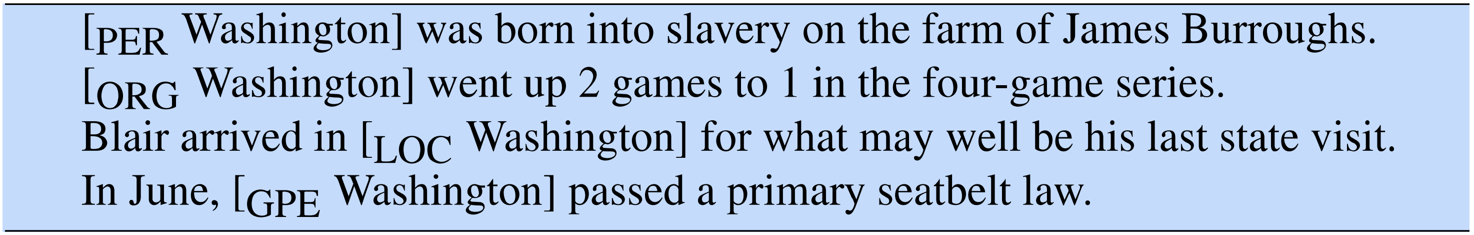
- The personal/location/organization are much more difficult to identified
  - E.g., “White House” can be either an organization or a location name in different context

# Named-Entity Recognition (2/4)

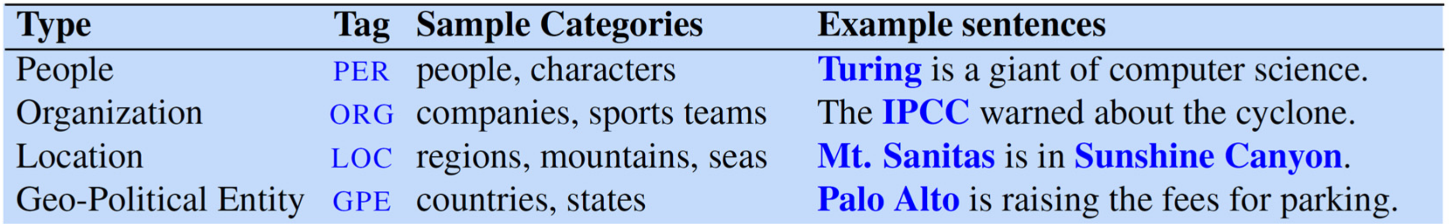
- Named-entity recognition (NER) has it origin from the Message Understanding Conferences (MUC) sponsored by U.S. DARPA program
  - Began in the 1990's
  - Aimed at extraction of information from text documents
  - Extended to many other languages and spoken documents (mainly broadcast news)
- Task Definition
  - A named entity is, roughly speaking, anything that can be referred to with a proper name: a person (**PER**), a location (**LOC**), an organization (**ORG**), etc.
  - NER is to find spans of text that constitute proper names and tag the type of the entity
- Common approaches to NER
  - Rule-based approach
  - Model-based approach
  - Combined approach

# Named-Entity Recognition (3/4)

- An example of the output of an NER tagger
  - There exist type ambiguities in the use of the name **Washington**



[$_{\text{PER}}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{\text{ORG}}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{\text{LOC}}$ Washington] for what may well be his last state visit.
In June, [$_{\text{GPE}}$ Washington] passed a primary seatbelt law.



| Type | Tag | Sample Categories | Example sentences |
|---|---|---|---|
| People | PER | people, characters | **Turing** is a giant of computer science. |
| Organization | ORG | companies, sports teams | The **IPCC** warned about the cyclone. |
| Location | LOC | regions, mountains, seas | **Mt. Sanitas** is in **Sunshine Canyon**. |
| Geo-Political Entity | GPE | countries, states | **Palo Alto** is raising the fees for parking. |

**Figure 8.5** A list of generic named entity types with the kinds of entities they refer to.

- Many applications will also need to use specific entity types like proteins, genes, commercial products, works of art, and others

# Named-Entity Recognition (4/4)

- Possible applications
    - In **sentiment analysis** we might want to know a consumer’s sentiment toward a particular entity
    - Entities are a useful first stage in **question answering**, or for linking text to information in structured knowledge sources like Wikipedia
    - Named entity tagging is also central to tasks involving **building semantic representations**, like extracting events and the relationship between participants
- Why NER is hard
    - Entity segmentation ambiguity
        - In POS tagging, no segmentation problem since each word gets one tag
        - In NER we have to find and segment the entities!
    - Entity type ambiguity

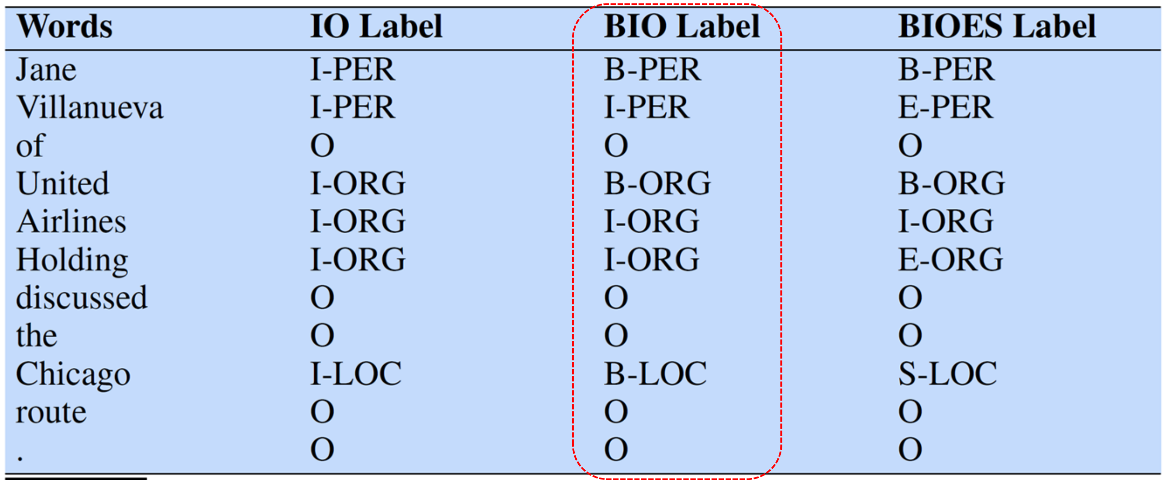
# BIO Tagging

- How can we turn this structured problem into a sequence problem like POS tagging, with one label per word?

[PER Jane Villanueva] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago ] route.

- BIO Tags
  - B: token that *begins* a span
  - I: tokens *inside* a span
  - O: tokens outside of any span

# of tags (where n is #entity types):
1 O tag, *n* B tags, *n* I tags
(total of *2n+1)*



| Words | IO Label | BIO Label | BIOES Label |
|---|---|---|---|
| Jane | I-PER | B-PER | B-PER |
| Villanueva | I-PER | I-PER | E-PER |
| of | O | O | O |
| United | I-ORG | B-ORG | B-ORG |
| Airlines | I-ORG | I-ORG | I-ORG |
| Holding | I-ORG | I-ORG | E-ORG |
| discussed | O | O | O |
| the | O | O | O |
| Chicago | I-LOC | B-LOC | S-LOC |
| route | O | O | O |
| . | O | O | O |

**Figure 8.7** NER as a sequence model, showing IO, BIO, and BIOES taggings.

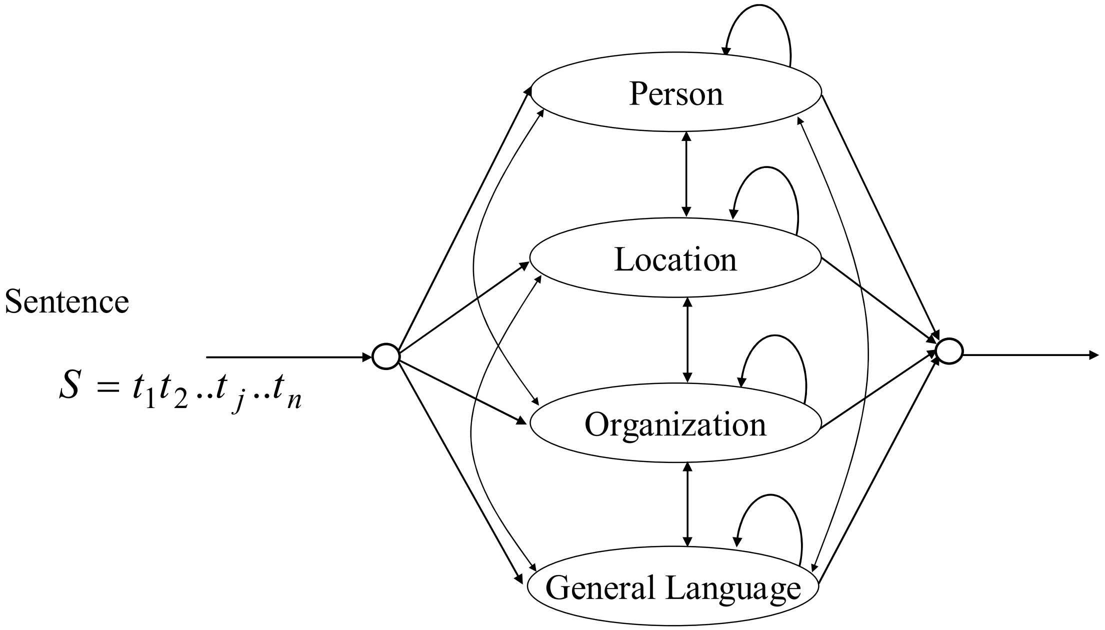
# NER: Rule-based Approach

- Employ various kinds of rules to identified named-entities; for example,
  - A cue-word “Co.” possibly indicates the existence of a company name in the span of its predecessor words
  - A cue-word “Mr.” possibly indicates the existence of a personal name in the span of its successor words

- However, the rules may become very complicated when we wish to cover all different possibilities
  - Time-consuming and difficult to handcraft all the rules
  - Especially when the task domain becomes more general, or when new sources of documents are being handled

# NER: Model-based Approach (1/2)

- The goal is usually to find the sequence of named entity labels (personal name, location name, etc.),$E = e_1 e_2 .. e_j .. e_n$, for a sentence, $S = t_1 t_2 .. t_j .. t_n$, which maximizes the probability $P(E|S)$
- For example, HMM is probably the best typical representative model used in this category



# NER: Model-based Approach (2/2)

- In HMM,
  - One state modeling each type of the named entities (person, location, organization)
  - One state modeling other words in the general language (non-named-entity words)
  - Possible transitions from states to states
  - Each state is characterized by a bi- or trigram language model
  - Viterbi search to find the most likely state sequence, or named entity label sequence $E$, for the input sentence, and the segment of consecutive words in the same named entity state is taken as a named entity

# NER: Combined approach

- For example, maximum entropy (ME) method
  - Many different linguistic and statistical features, such as part-of-speech (POS) information, rule-based knowledge, term frequencies, etc., can all be represented and integrated in this method
  - It was shown that very promising results can be obtained with this method

# NER: OOV words (1/4)

- Handling out-of-vocabulary (OOV) or unknown words
  - E.g., HMM
    - Divide the training data into two parts during training
      - In each half, every segment of terms or words that does not appear in the other half is marked as “Unknown”, such that the probabilities for both known and unknown words occurring in the respective named-entity states can be properly estimated
    - During testing, any segment of terms that is not seen before can thus be labeled “Unknown,” and the Viterbi algorithm can be carried out to give the desired results

# NER: OOV words (2/4)

- Handling out-of-vocabulary (OOV) or unknown words for spoken documents
    - Out-of-vocabulary (OOV) problem is raised due to the limitation in the vocabulary size of speech recognizer
    - OOV words will be misrecognized as other in-vocabulary words
        - Lose their true semantic meanings
- Tackle this problem using ASR & IR techniques
    - In ASR (automatic speech recognition)
        - Spoken docs are transcribed using a recognizer implemented with a lexical network modeling both word- and subword-level (phone or syllable) *n*-gram LM constraints
            - The speech portions corresponding to OOV words may be properly decoded into sequences of subword units

# NER: OOV words (3/4)

- Tackle this problem using ASR & IR techniques
    - The subword *n*-gram LM is trained by the text segments corresponding to the low-frequency words not included in the vocabulary of the recognizer
  - In IR (Information Retrieval)
    - A retrieval process was performed using each spoken doc itself as a query to retrieve relevant docs from a temporal/topical homogeneous reference text collection
    - The indexing terms adopted here can be either word-level features, subword-level features, or both of them

# NER: OOV words (4/4)

- Tackle this problem using ASR & IR techniques
  - Once the top-ranked text documents are selected, each decoded subword sequence within the spoken document, that are corresponding to a possible OOV word, can be used to match every possible text segments or word sequences within the top-ranked text documents
  - The text segment or word sequence within the top-ranked text docs that has the maximum combined score of phonetic similarity to the OOV word and relative frequency in the relevant text docs can thus be used to replace the decoded subword sequence of the spoken document

$$\max_{w} \sum_{d\in D_r} P(e_{oov}|w)\cdot P(w|d)\cdot P\big(d\big|q_s\big)$$

phone/syllable sequence of the OOV words

word in the top-ranked relevant text document set

spoken document belonging to the top-ranked relevant text document set