Lecture 24: Part of Speech Tagging

Mark Hasegawa-Johnson

3/2022

Luca della Robbia, *Priscian, or the Grammarian* (1437-1439). Marble panel from the North side, lower basement of the bell tower of Florence, Italy. Museo dell'Opera del Duomo. Public domain photo by Jastrow, 2006

Outline

- Syntax and semantics
- Part of speech tagging
- An HMM for POS tagging
- The Viterbi algorithm

Semantics: Montague grammar

Richard Montague, 1930-1971
photograph © Richard Thomason

Richard Montague defined formal semantics as follows:

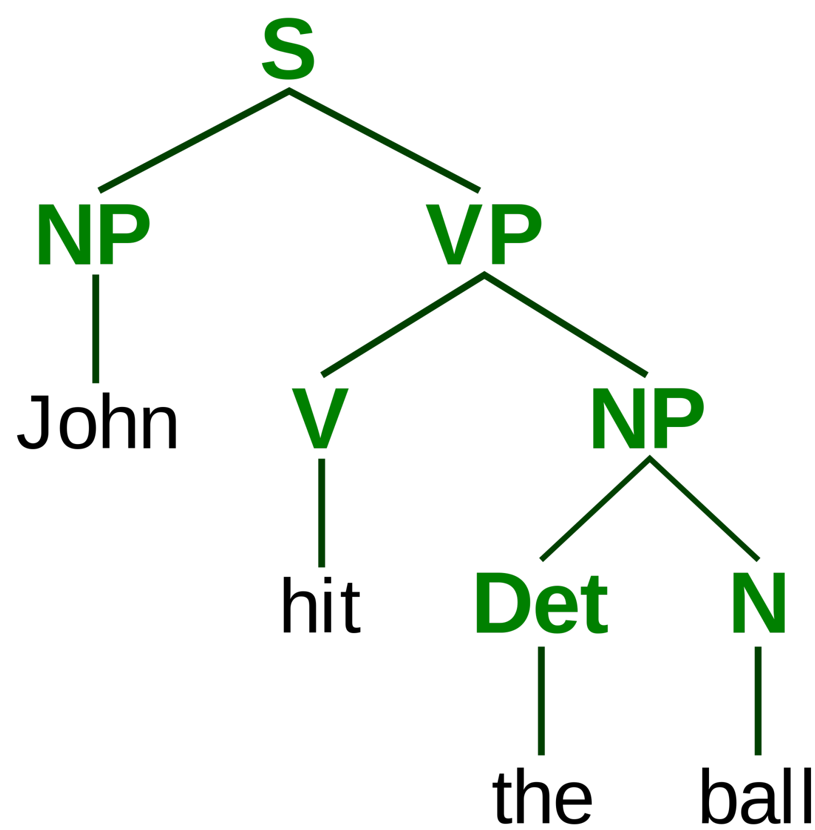
- ”Understanding a sentence” means that you can specify the conditions under which the sentence would be true
- The meaning of a sentence is composed of the meanings of its words. For example:

Logical form	Example	Meaning
some(P,Q)	“some people sing”	$\exists x: ((Px) \land (Qx))$
a(P,Q)	“a bird sings”	$\exists x: ((Px) \land (Qx))$
every(P,Q)	“every bird sings”	$\forall x: ((Px) \rightarrow (Qx))$
no(P,Q)	“no bird snores”	$\forall x: ((Px) \rightarrow \neg(Qx))$

Syntax

Syntax is the study of how words combine.

- Syntax is a descriptive science: it simply describes how words combine when people are using them naturally.
- Compositional semantics studies the meanings of those combinations.

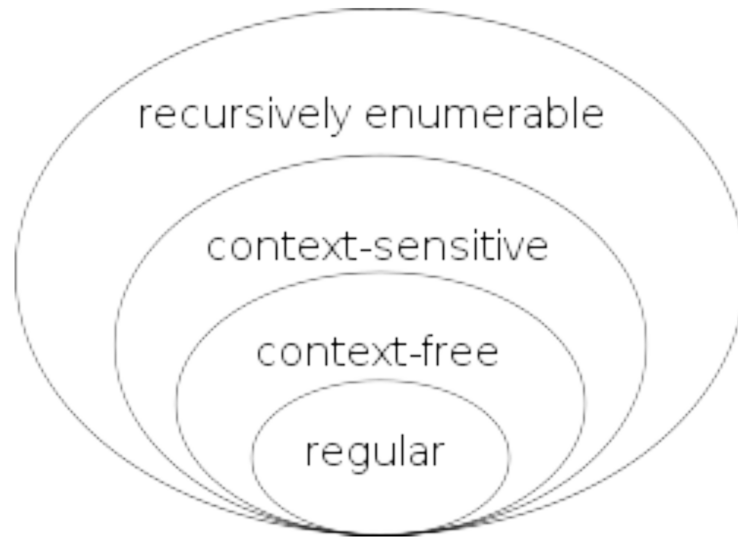


ParseTree.svg. Public domain image, Stannered, 2007

Grammar

A **grammar** is a mathematical specification of the set of all word sequences that form valid sentences in a language (e.g., English).

- **Recursively enumerable**: any grammar that can be decided by a Turing machine
- **Context-sensitive**: phrase A is expanded into phrases B and C using rules of the form $\alpha A\beta \to \alpha BC\beta$ for specified contexts α and β.
- **Context-free**: phrase A is expanded into phrases B and C using context-free rules: $A \to BC$.
- **Regular**: phrase A can only be expanded into a word followed by another phrase: $A \to aB$.

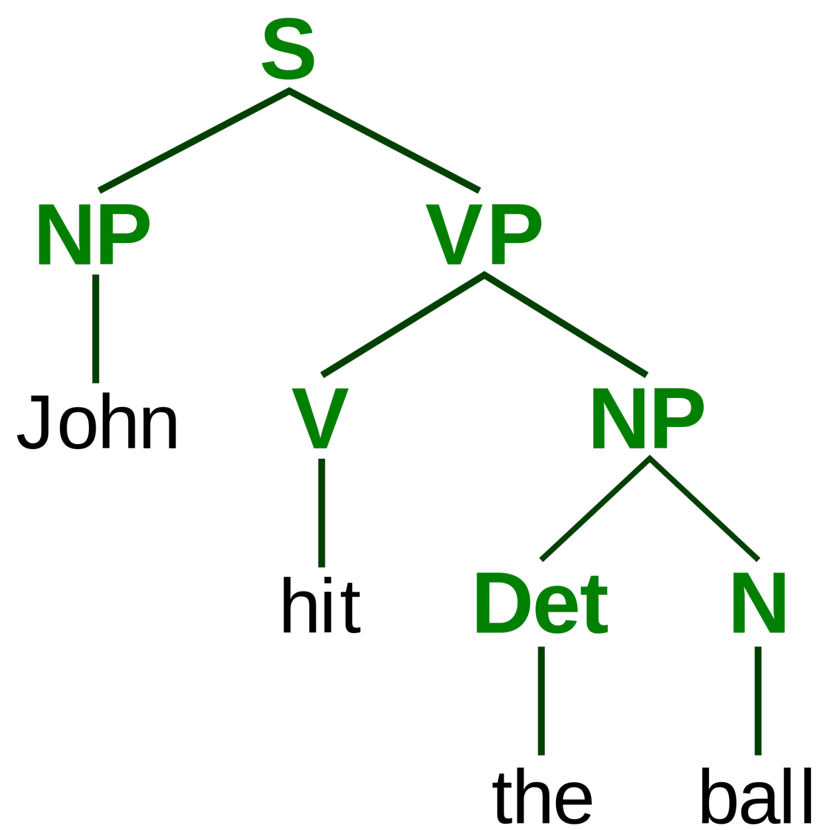


Chomsky-hierarchy.svg. CC-SA 3.0, J. Finkelstein, 2010

Grammar

Humans usually think of natural language using context-free grammar (CFG). For example,

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP$$
$$NP \rightarrow Det\ N$$
$$NP \rightarrow \text{John}$$
$$V \rightarrow \text{hit}$$
$$Det \rightarrow \text{the}$$
$$N \rightarrow \text{ball}$$

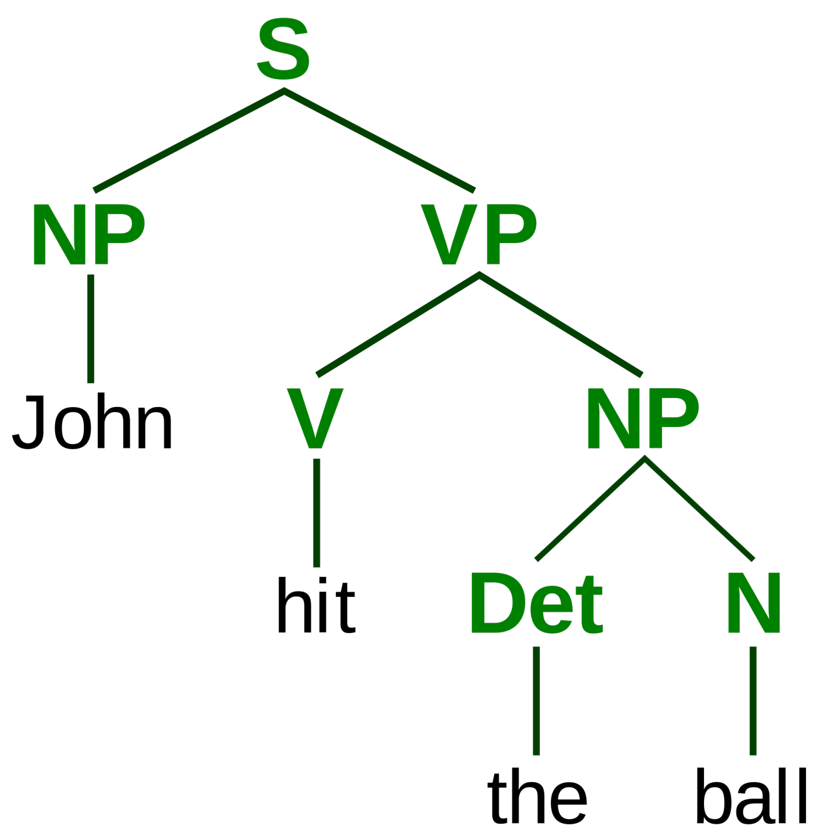


ParseTree.svg. Public domain image, Stannered, 2007

Grammar

A CFG with finite recursion depth can be written as a regular grammar. For example:

$$S \rightarrow \text{John}\, VP$$
$$VP \rightarrow \text{hit}\, NP$$
$$NP \rightarrow \text{the}\, N$$
$$N \rightarrow \text{ball}$$



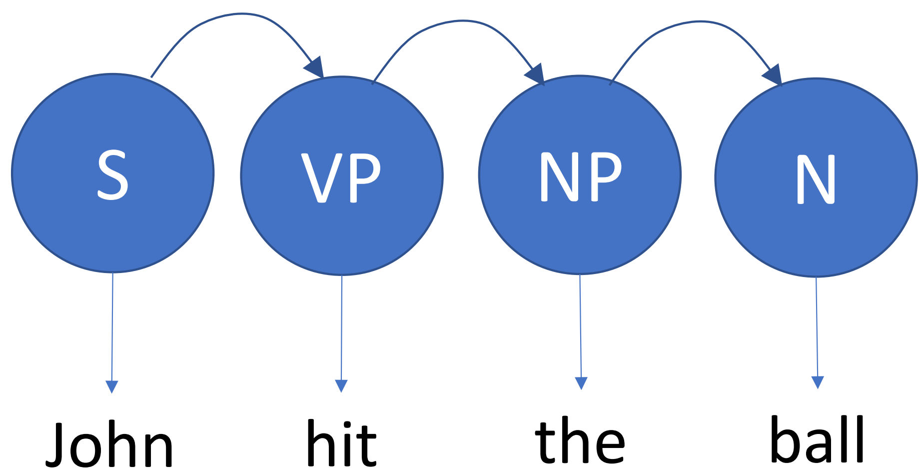
ParseTree.svg. Public domain image, Stannered, 2007

Grammar

A regular grammar can be written using an HMM.

- The phrase is the state variable
- The word is the observed variable

$$S \rightarrow \text{John } VP$$
$$VP \rightarrow \text{hit } NP$$
$$NP \rightarrow \text{the } N$$
$$N \rightarrow \text{ball}$$



Key concepts: syntax and semantics

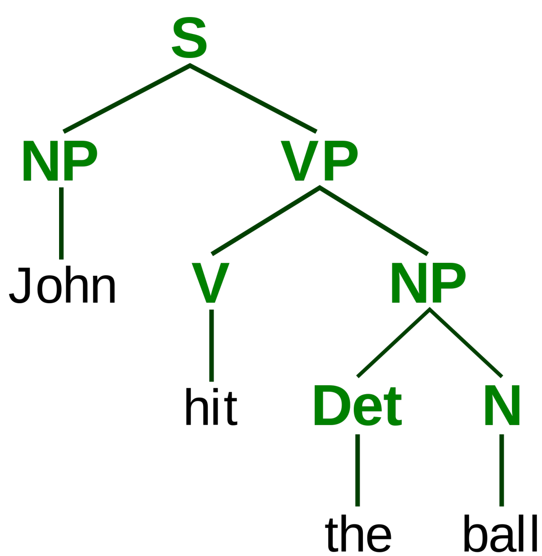
- Compositional semantics studies how sentence meaning is computed from word meanings.
- Syntax studies the ways in which words combine.
- A grammar is a mathematical specification of the sequences of words that form valid sentences in a language.
- A context-free grammar with finite recursion depth can be written as a regular grammar.
- A regular grammar can be written as an HMM.

Outline

- Syntax and semantics
- Part of speech tagging
- An HMM for POS tagging
- The Viterbi algorithm

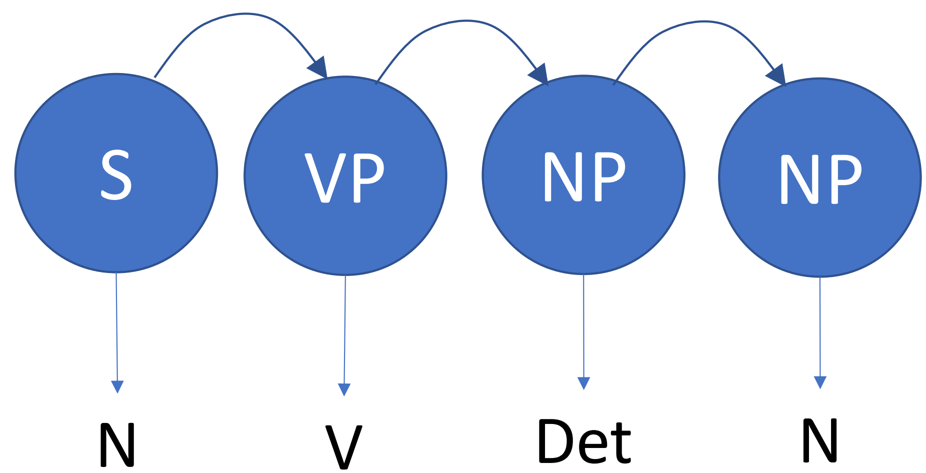
Parts of speech

Many grammars are written in terms of parts of speech, to make them a bit more general. For example, this one…



ParseTree.svg. Public domain image, Stannered, 2007

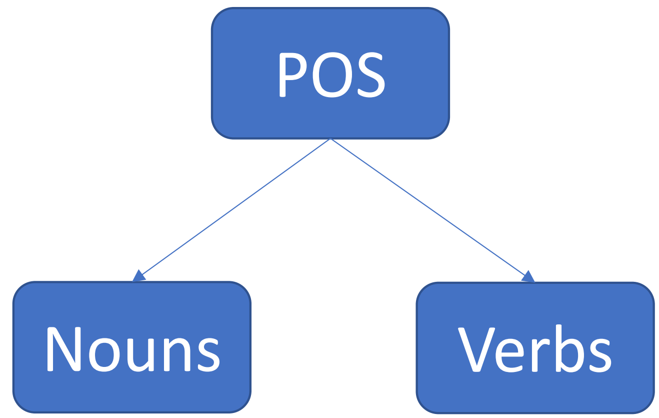
…could be generalized like this…



Parts of speech

For some reason, most of the part-of-speech (POS) systems proposed by philosophers have 2^N parts of speech, for some value of N.

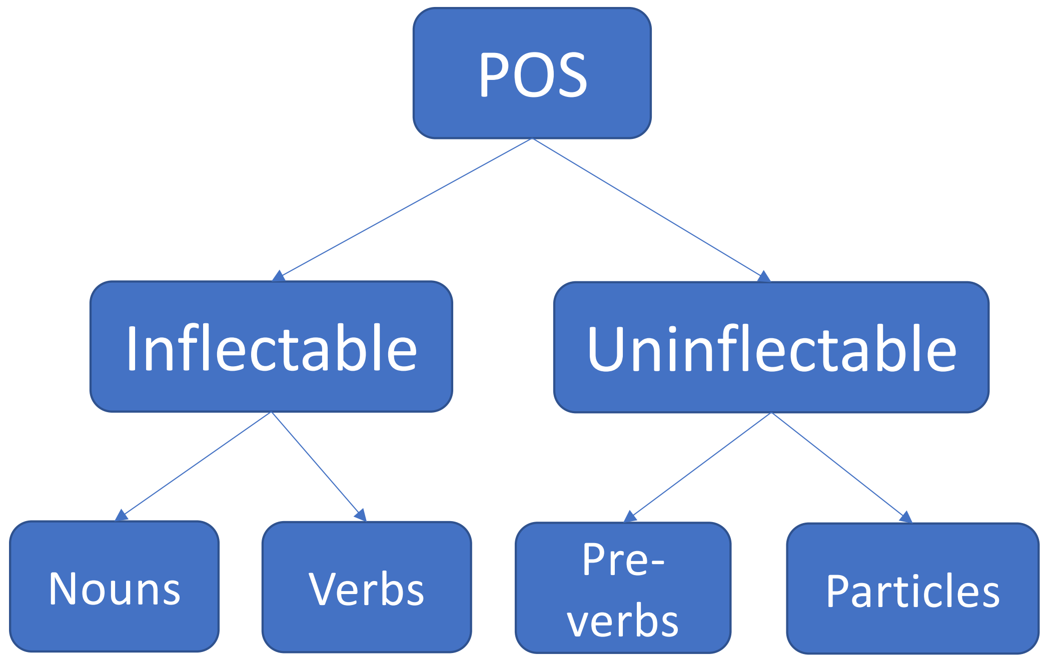
- Plato (350BC) proposed that there are 2 parts of speech: nouns and verbs.



Parts of speech

For some reason, most of the part-of-speech (POS) systems proposed by philosophers have 2^N parts of speech, for some value of N.

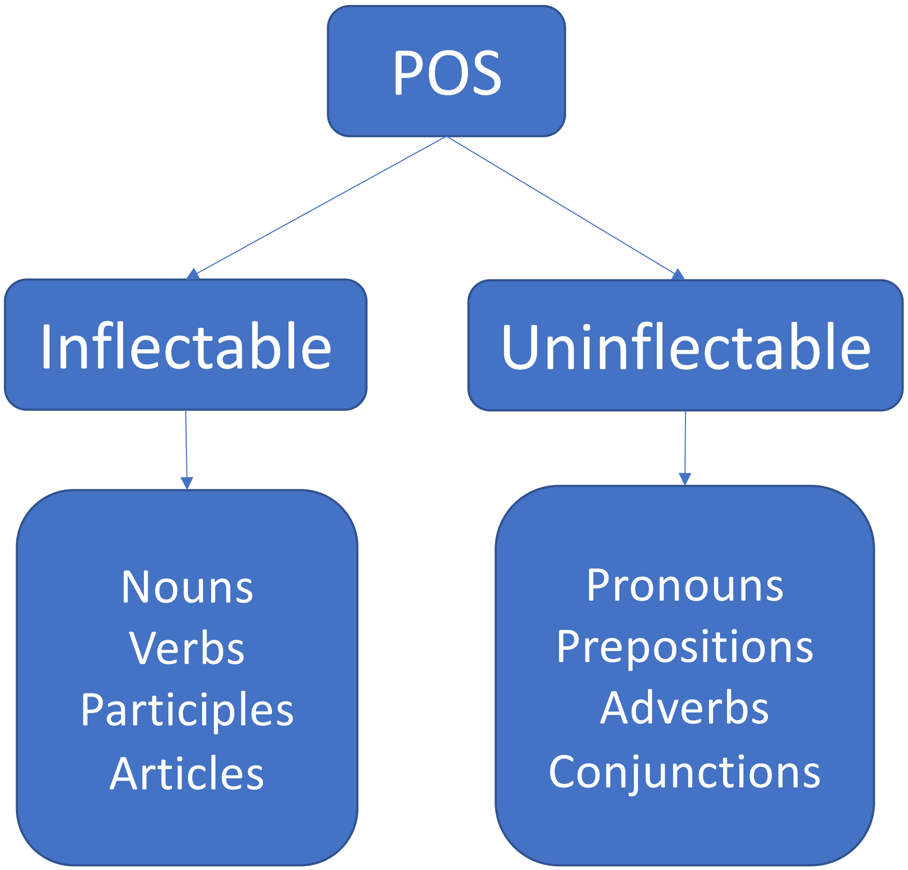
- Plato (350BC) proposed that there are 2 parts of speech: nouns and verbs.
- Yāska (600BC) proposed that there are 4 parts of speech.



Parts of speech

For some reason, most of the part-of-speech (POS) systems proposed by philosophers have 2^N parts of speech, for some value of N.

- Plato (350BC) proposed that there are 2 parts of speech: nouns and verbs.
- Yāska (600BC) proposed that there are 4 parts of speech.
- Dionysus Thrax (100BC) proposed 8 parts of speech.



Parts of speech

Most modern English dictionaries use these POS tags.

- **Open-class words** (anybody can make up a new word, in any of these classes, at any time): nouns, verbs, adjectives, adverbs, interjections
- **Closed-class words** (it's hard to make up a new word in these classes): pronouns, prepositions, conjunctions, determiners

Most published, tagged data use POS tags that are finer-grained than the nine tags listed above. For example, the next few slides describe the Penn Treebank POS tag set.

Nouns

The Penn Treebank noun categories are:

- NN (singular or mass common noun): llama, thought, communism
- NNS (plural common noun): llamas, thoughts
- NNP (singular proper noun): Jane, IBM, Mexico
- NNPS (plural proper noun): Osbournes, Carolinas
- VBG (gerund): eating

Verbs

The Penn Treebank verb categories are:

- VB (verb base form): eat
- VBD (verb past tense): ate
- VBP (verb non-3sg present): eat
- VBZ (verb 3sg present): eats
- MD (modal): can as in “can lift”, should as in “should go”
- RP (particle): up as in “get up,” off as in “take off”

Adjectives

The Penn Treebank has several categories that might be considered types of adjectives:

- CD (cardinal number --- use this tag regardless of whether the number is being used as a noun or adjective): one, two, twenty
- JJ (adjective): yellow, exceptional, tall
- JJR (comparative adjective): yellower, taller
- JJS (superlative adjective): yellowest, tallest
- PRP$ (possessive pronoun): your, one’s
- VBN (verb past participle): eaten, compiled
- WP$ (wh-possessive): whose

Determiners, Prepositions and Conjunctions

The Penn Treebank has a lot of things that look like determiners, prepositions, or conjunctions:

- CC (coordinating conjunction): and, but, or
- DT (determiner): a, the
- IN (preposition or subordinating conjunction): of, in, by
- PDT (predeterminer): all, both
- POS (possessive ending): ‘s, as in “Bob’s dog”
- TO (any use of the word “to”): to
- WDT (wh-determiner): which, that

Why do POS tagging?

- Because it's highly accurate, typically 97%. That means you can run a POS tagger as a pre-processing step, before doing harder natural language understanding tasks.
- Because it's necessary, if you want to know what the words in the sentence mean.

Will Will ? Will will . Will will will Will 's will to Will .

MD NNP SYM NNP MD SYM NNP MD VB NNP POS NN TO NNP SYM

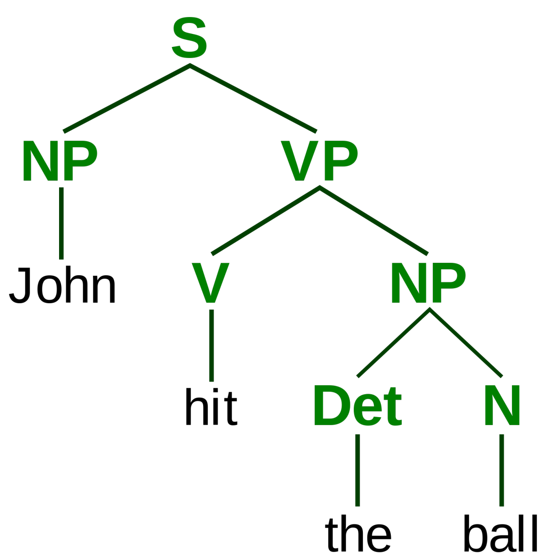
Outline

- Syntax and semantics
- Part of speech tagging
- An HMM for POS tagging
- The Viterbi algorithm

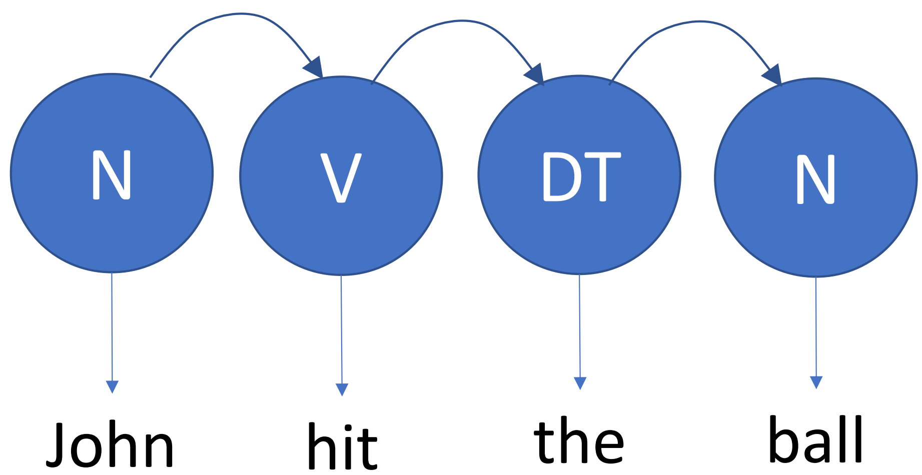
An HMM for POS tagging

The basic idea of an HMM POS tagger is:

- Treat the part of speech as the hidden state variable
- Treat the word as observed



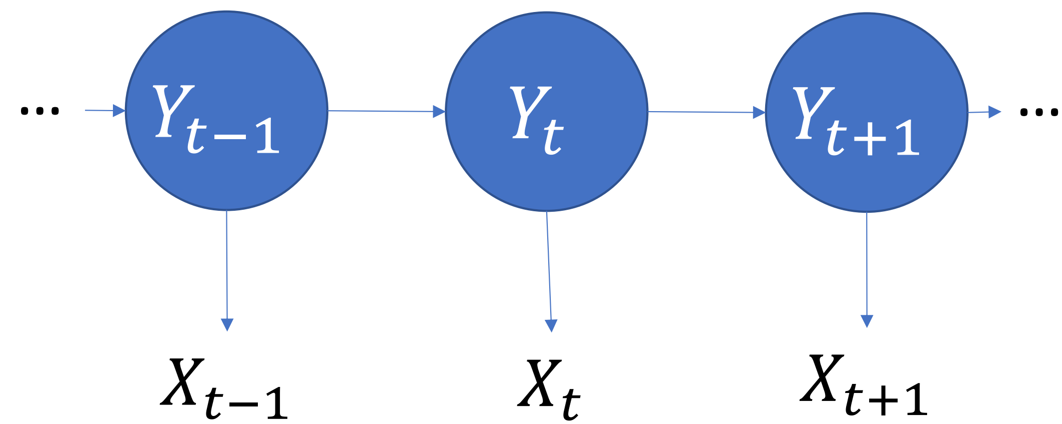
ParseTree.svg. Public domain image, Stannered, 2007



HMM as a Bayes Net

This slide shows an HMM as a Bayes Net. You should remember the graph semantics of a Bayes net:

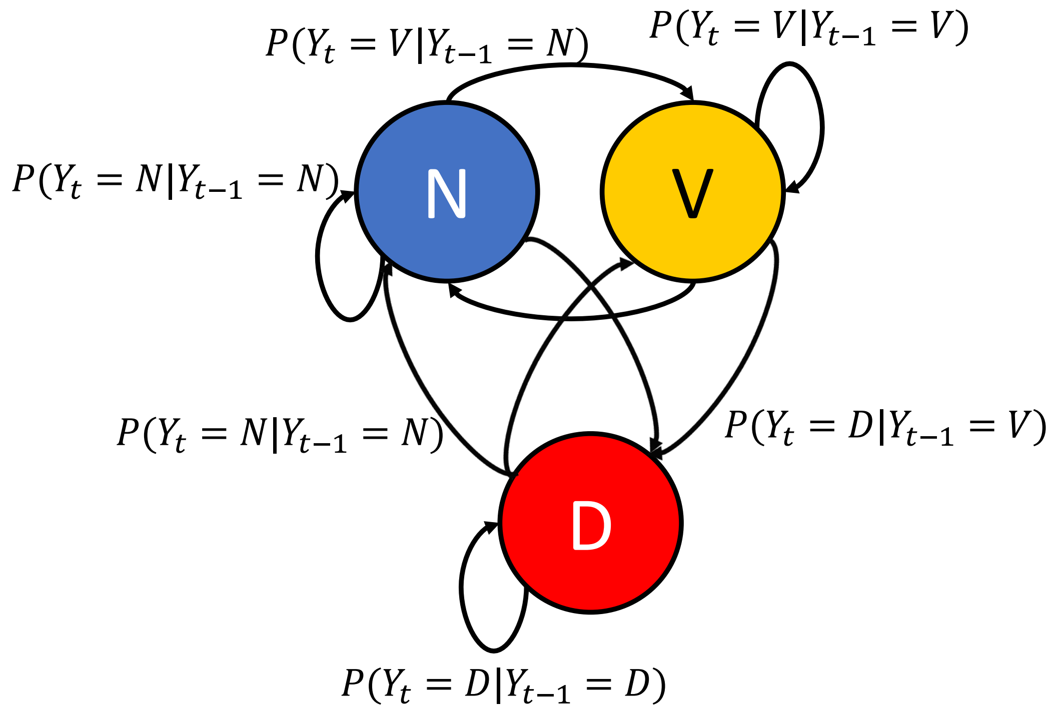
- Nodes are random variables.
- Edges denote stochastic dependence.



HMM as a Finite State Machine

This slide shows ***exactly the same HMM,*** viewed in a totally different way. Here, we show it as a finite state machine:

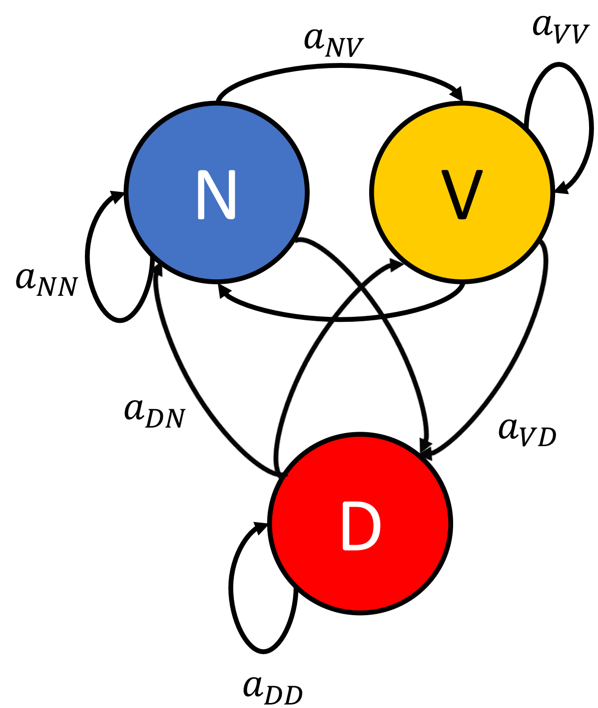
- Nodes denote states.
- Edges denote possible transitions between the states.



Parameters of an HMM

Suppose that there are N distinct POS tags, and V distinct words. Then the parameters of an HMM are:

- $\pi_j = P(Y_1 = j)$. There are N of these.
- $a_{ij} = P(Y_t = j | Y_{t-1} = i)$. There are N^2 of these.
- $b_{jk} = P(X_t = k | Y_t = j)$. There are NV of these.

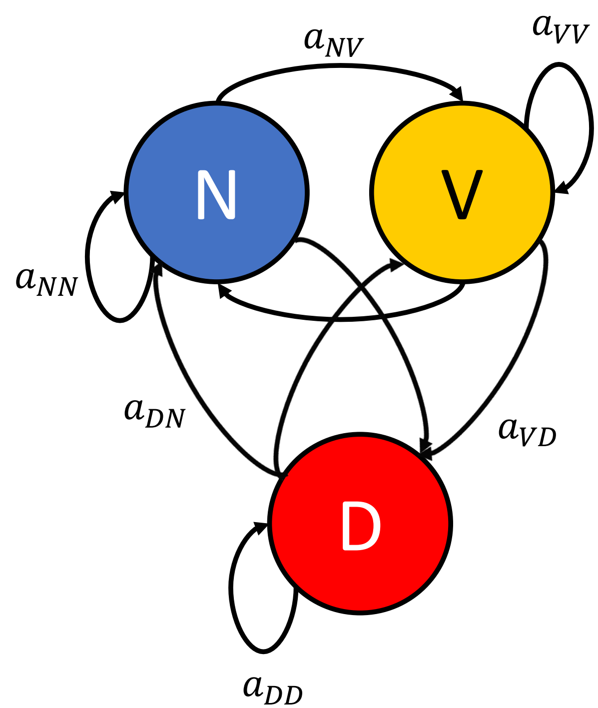


Estimating the Parameters of an HMM

$$\pi_j = \frac{(\#\text{sentences that start with POS } j) + k}{(\#\text{sentences in the training corpus}) + kN}$$

$$a_{ij} = \frac{(\#\text{times } j \text{ follows } i) + k}{(\#\text{times tag } i \text{ occurs in training corpus}) + kN}$$

$$b_{jk} = \frac{(\#\text{times tag } j \text{ is matched to word } k) + k}{(\#\text{times tag } j \text{ occurs in training corpus}) + k(V+1)}$$



Outline

- Syntax and semantics
- Part of speech tagging
- An HMM for POS tagging
- The Viterbi algorithm

Viterbi Algorithm: Key concepts

Nodes and edges have numbers attached to them:

- **Edge Probability**: Probability of taking that transition, and then generating the next observed output

$$e_{ijt} = P(Y_t = j, X_t = x_t | Y_{t-1} = i)$$

- **Node Probability**: Probability of the best path until node j at time t

$$v_{j,t} = \max_{y_1,\ldots,y_{t-1}} P(X_1 = x_1 \ldots, X_t = x_t, Y_1 = y_1, \ldots, Y_t = j)$$

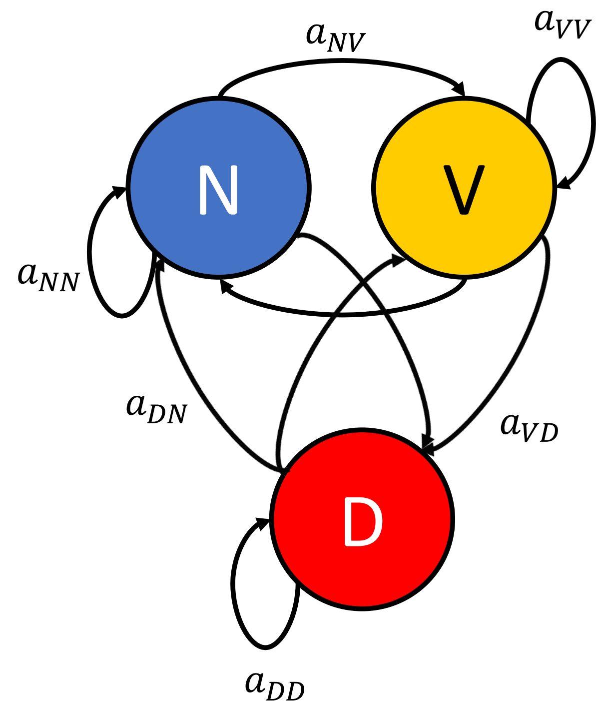
Viterbi Algorithm for POS tagging

- **Edge Probability**:

$$
\begin{aligned}
e_{ijt} &= P(Y_t = j, X_t = x_t | Y_{t-1} = i) \\
&= P(Y_t = j | Y_{t-1} = i) P(X_t = x_t | Y_t = j) \\
&= a_{ij} b_{j,x_t}
\end{aligned}
$$

- **Initial Node Probability**: Probability of starting in a particular node:

$$
\begin{aligned}
v_{j,1} &= P(X_1 = x_1, Y_1 = j) \\
&= P(Y_1 = j) P(X_t = x_1 | Y_t = j) \\
&= \pi_j b_{j,x_1}
\end{aligned}
$$



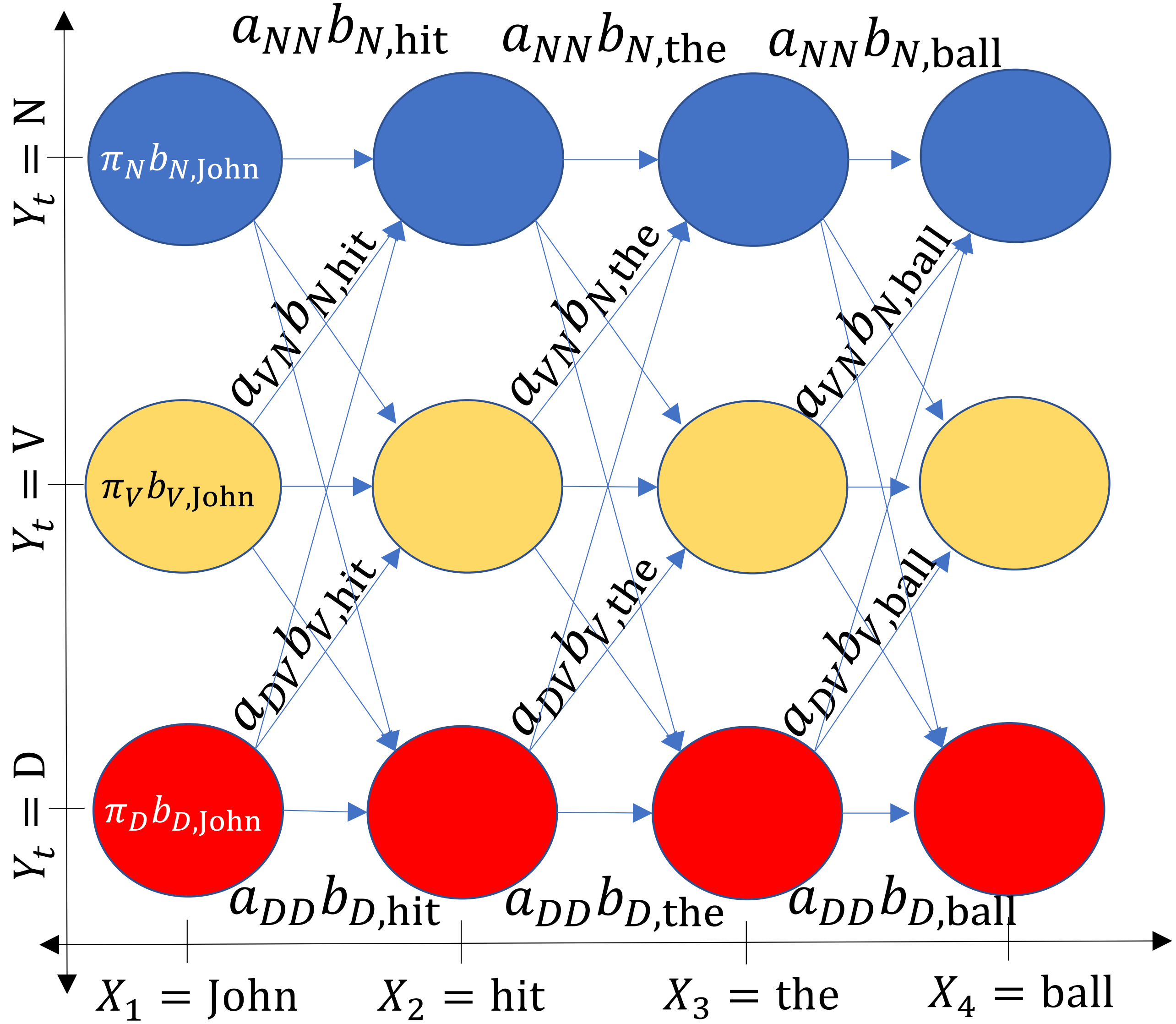
Trellis

Initial Node Probability:

$$v_{j,1} = \pi_j b_{j,x_1}$$

Edge Probability:

$$e_{ijt} = a_{ij} b_{j,x_t}$$



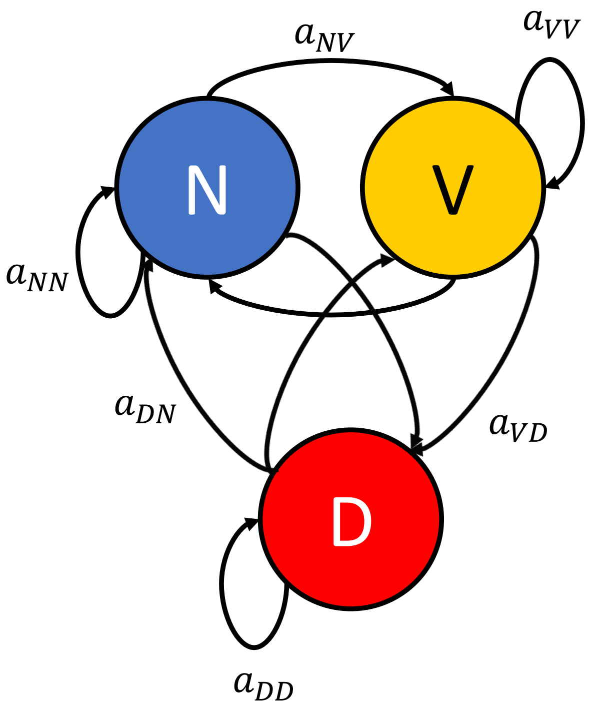
Viterbi Algorithm for POS tagging

<u>**Node Probability**</u>:

$$v_{j,t} = \max_{y_1,\ldots,y_{t-1}} P(\ldots, X_t = x_t, \ldots, Y_t = j)$$

$$= \max_i \left(\max_{y_1,\ldots,y_{t-2}} P(\ldots, X_{t-1} = x_{t-1}, \ldots, Y_{t-1} = i) P(Y_t = j | Y_{t-1} = i) P(X_t = x_t | Y_t = j) \right)$$

$$= \max_i v_{i,t-1} e_{ijt}$$



Trellis

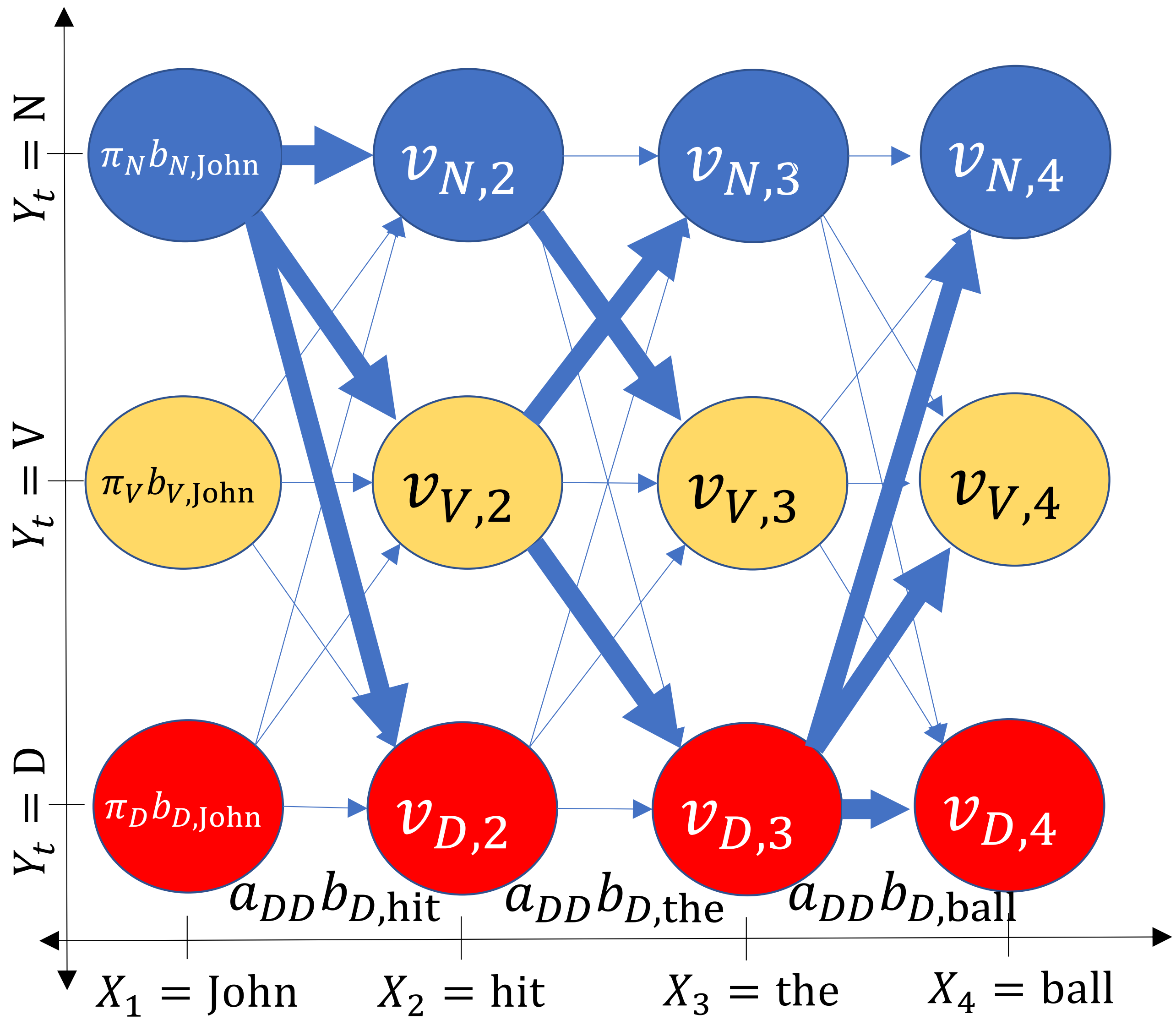
Node Probability:

$$v_{j,t} = \max_i v_{i,t-1} e_{ijt}$$

Backpointer:

$$i^*_{j,t} = \operatorname{argmax}_i v_{i,t-1} e_{ijt}$$

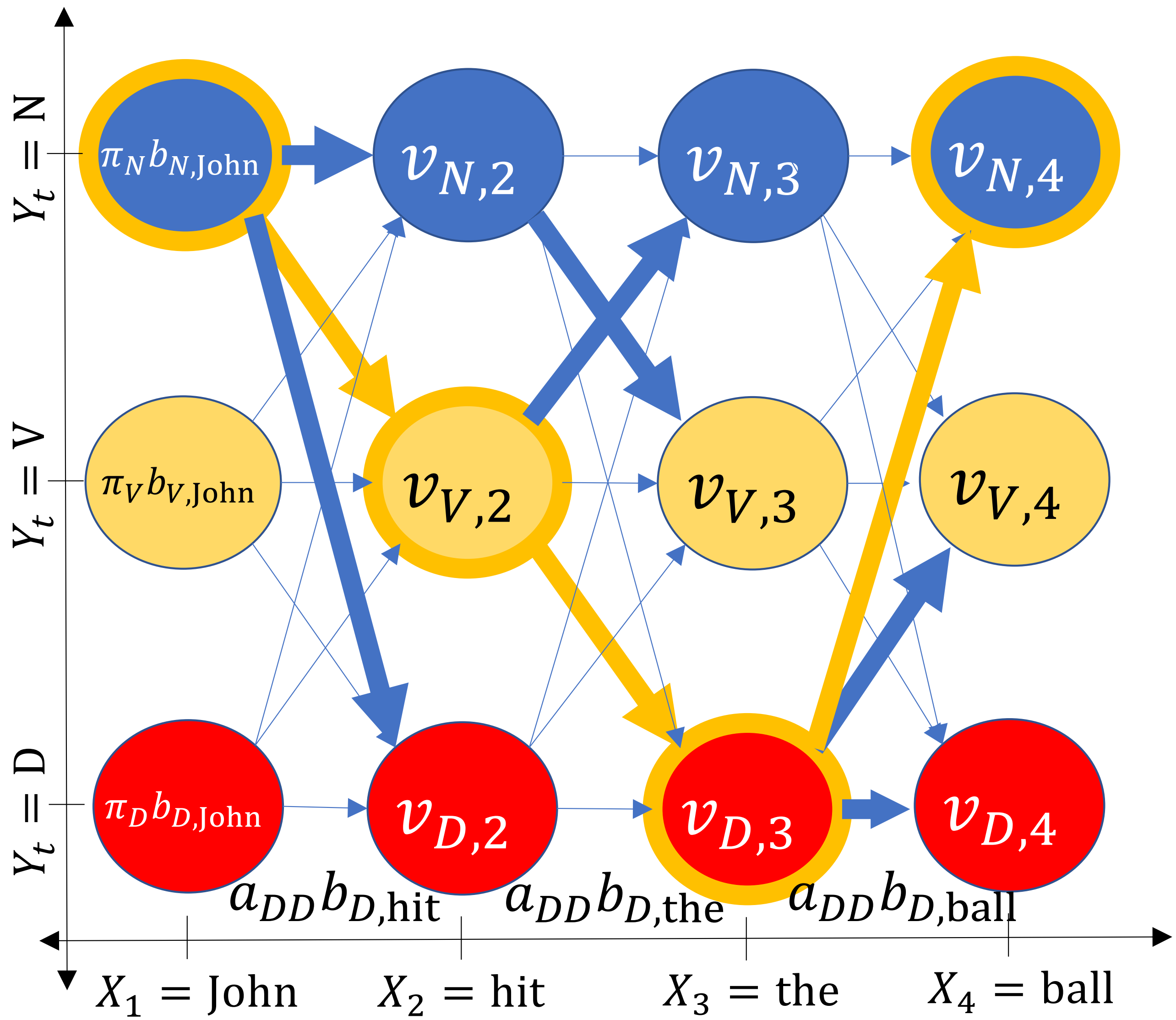
Shown: possible backpointers. Actual backpointers depend on model parameters!



Backtrace

Find the node with the highest value of $v_{j,T}$ at the end, and follow the backpointers!

Shown: possible backtrace. Actual backtrace depends on model parameters!



Viterbi algorithm key formulas

Initial Node Probability:

$$v_{j,1} = \pi_j b_{j,x_1}$$

Edge Probability:

$$e_{ijt} = a_{ij} b_{j,x_t}$$

Node Probability:

$$v_{j,t} = \max_i v_{i,t-1} e_{ijt}$$

Backpointer:

$$i^*_{j,t} = \operatorname*{argmax}_i v_{i,t-1} e_{ijt}$$

Viterbi algorithm key formulas

Initial Node Probability:

$$\log v_{j,1} = \log \pi_j + \log b_{j,x_1}$$

Edge Probability:

$$\log e_{ijt} = \log a_{ij} + \log b_{j,x_t}$$

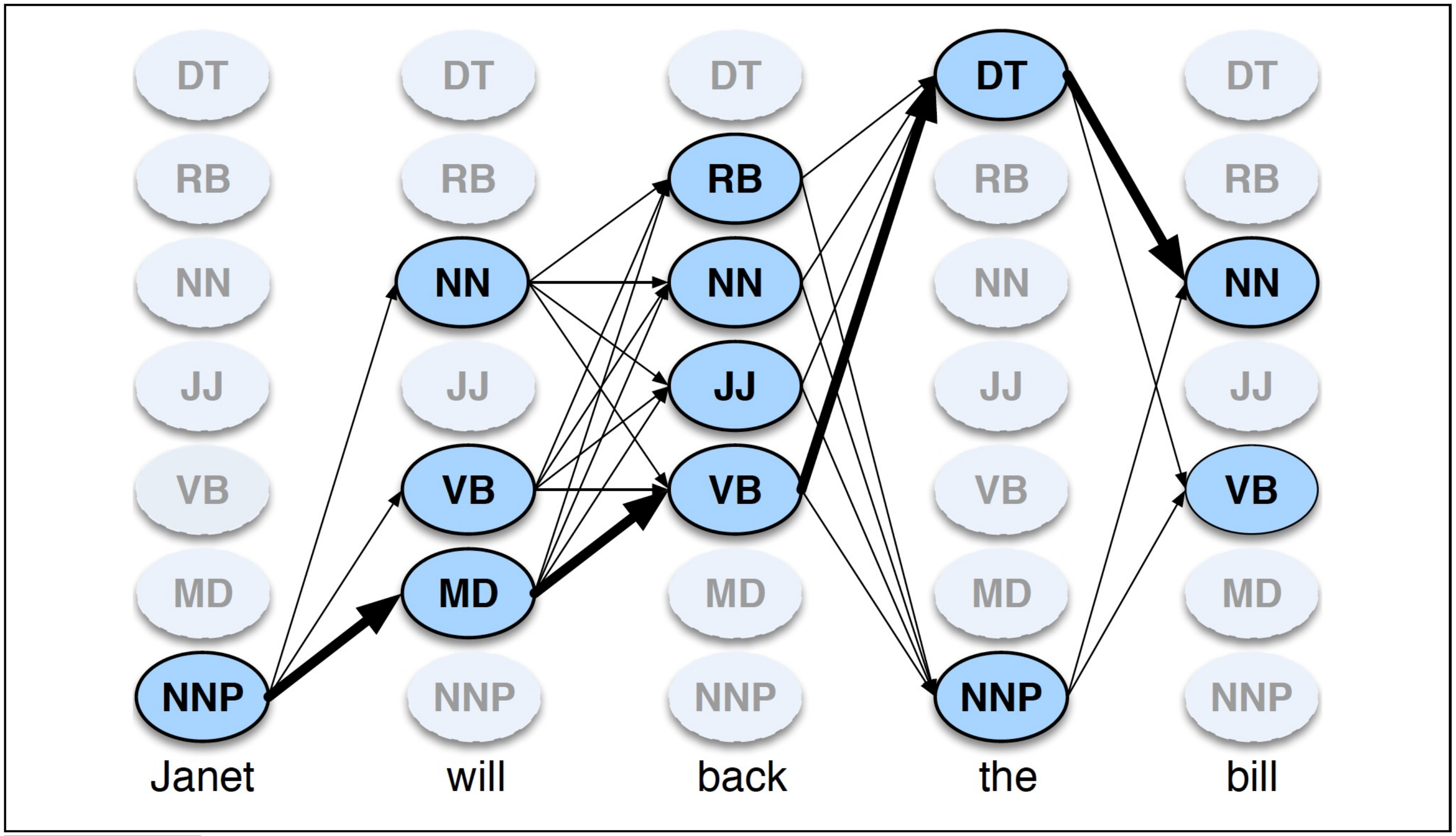
Node Probability:

$$\log v_{j,t} = \max_i \left(\log v_{i,t-1} + \log e_{ijt}\right)$$

Backpointer:

$$i^*_{j,t} = \operatorname*{argmax}_i \left(\log v_{i,t-1} + \log e_{ijt}\right)$$

Example from Jurafsky & Martin



Outline

- Syntax and semantics
  - A grammar specifies which word sequences are valid sentences
  - Finite-depth CFG = Regular grammar = HMM
- Part of speech tagging
  - Open-class words: nouns, verbs, adverbs, adjectives, interjections
  - Closed-class words: prepositions, pronouns, conjunctions, determiners
- An HMM for POS tagging
  - State variable is the part of speech
  - Observation is the word
- The Viterbi algorithm
  - $\log v_{j,t} = \max_i \left(\log v_{i,t-1} + \log e_{ijt} \right)$