

Lecture 4:

Sequence Methods

Part 1: Part of Speech Tagging and
Hidden Markov Models

Part of Speech (POS) Tagging

INPUT:

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:

Profits/**N** soared/**V** at/**P** Boeing/**N** Co./**N** ,/, easily/**ADV** topping/**V**
forecasts/**N** on/**P** Wall/**N** Street/**N** ,/, as/**P** their/**POSS** CEO/**N**
Alan/**N** Mulally/**N** announced/**V** first/**ADJ** quarter/**N** results/**N** ./.

N = Noun

V = Verb

P = Preposition

Adv = Adverb

Adj = Adjective

...

Part of Speech (POS) Tagging

Training set:

1 Pierre/**NNP** Vinken/**NNP** ,/, 61/**CD** years/**NNS** old/**JJ** ,/, will/**MD** join/**VB** the/**DT** board/**NN** as/**IN** a/**DT** nonexecutive/**JJ** director/**NN** Nov./**NNP** 29/**CD** ./.

2 Mr./**NNP** Vinken/**NNP** is/**VBZ** chairman/**NN** of/**IN** Elsevier/**NNP** N.V./**NNP** ,/, the/**DT** Dutch/**NNP** publishing/**VBG** group/**NN** ./.

3 Rudolph/**NNP** Agnew/**NNP** ,/, 55/**CD** years/**NNS** old/**JJ** and/**CC** chairman/**NN** of/**IN** Consolidated/**NNP** Gold/**NNP** Fields/**NNP** PLC/**NNP** ,/, was/**VBD** named/**VBN** a/**DT** nonexecutive/**JJ** director/**NN** of/**IN** this/**DT** British/**JJ** industrial/**JJ** conglomerate/**NN** ./.

...

38,219 It/**PRP** is/**VBZ** also/**RB** pulling/**VBG** 20/**CD** people/**NNS** out/**IN** of/**IN** Puerto/**NNP** Rico/**NNP** ,/, who/**WP** were/**VBD** helping/**VBG** Hurricane/**NNP** Hugo/**NNP** victims/**NNS** ,/, and/**CC** sending/**VBG** them/**PRP** to/**TO** San/**NNP** Francisco/**NNP** instead/**RB** ./.

- From the training set, induce a function/algorithm that maps new sentences to their tag sequences.

Tagging – the Supervised Setup

Training set:

1 Pierre/**NNP** Vinken/**NNP** ,/, 61/**CD** years/**NNS** old/**JJ** ,/, will/**MD** join/**VB** the/**DT** board/**NN** as/**IN** a/**DT** nonexecutive/**JJ** director/**NN** Nov./**NNP** 29/**CD** ./.

2 Mr./**NNP** Vinken/**NNP** is/**VBZ** chairman/**NN** of/**IN** Elsevier/**NNP** N.V./**NNP** ,/, the/**DT** Dutch/**NNP** publishing/**VBG** group/**NN** ./.

3 Rudolph/**NNP** Agnew/**NNP** ,/, 55/**CD** years/**NNS** old/**JJ** and/**CC** chairman/**NN** of/**IN** Consolidated/**NNP** Gold/**NNP** Fields/**NNP** PLC/**NNP** ,/, was/**VBD** named/**VBN** a/**DT** nonexecutive/**JJ** director/**NN** of/**IN** this/**DT** British/**JJ** industrial/**JJ** conglomerate/**NN** ./.

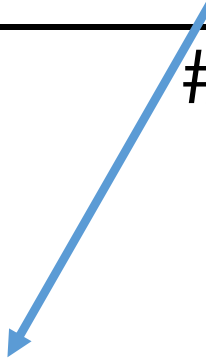
...

38,219 It/**PRP** is/**VBZ** also/**RB** pulling/**VBG** 20/**CD** people/**NNS** out/**IN** of/**IN** Puerto/**NNP** Rico/**NNP** ,/, who/**WP** were/**VBD** helping/**VBG** Hurricane/**NNP** Hugo/**NNP** victims/**NNS** ,/, and/**CC** sending/**VBG** them/**PRP** to/**TO** San/**NNP** Francisco/**NNP** instead/**RB** ./.

Evaluation:

accuracy

$$= \frac{\# \text{ test set words with correct tag}}{\# \text{ test set words}}$$



Word tokens as opposed to *word types*. That is of the *word type* dog appears 5 times in the test set, it will be counted 5 times by the accuracy measure

Named Entity Recognition (NER)

INPUT: Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT: Profits soared at [Company Boeing Co.], easily topping forecasts on [Location Wall Street], as their CEO [Person Alan Mulally] announced first quarter results.

NER as a Sequence Labeling Task

INPUT:

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA
topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA
their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA
quarter/NA results/NA ./NA

NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location

...

Task Definition: Part of Speech Tags

- Parts of Speech are word categories that have similar morphological or syntactic behavior
- For instance, we can define an adjective as a word which appears as a modifier in a Noun Phrase, as a predicate in a copula clause or in a comparative construction
 - The _____ dog (*black/red/hungry/tall*)
 - The dog is _____ (*black/red/hungry/tall*)
 - X / X+er / X + est (*black/red/hungry/tall*)
- **Circular definition! syntactic environments are defined in terms of grammatical categories**

Part of Speech Tags

- Wordforms often have more than one possible POS: *back*
 - The *back* door = *Adj*
 - On my *back* = *Noun*
 - Win the voters *back* = *Adverb*
 - Promised to *back* the bill = *Verb*
- The POS tagging problem is to determine the (single) POS tag for a particular word token (instance)

Sources of information

- What are the main sources of information for POS tagging?
 1. Knowledge of word probabilities
 - *man* is rarely used as a verb...
 2. Knowledge of neighboring words

Bill	saw	that	man	yesterday
name	verb (past)	det.	noun	adverb
<i>verb</i>	<i>verb</i>	<i>det.</i>	<i>verb</i>	<i>adverb</i>
<i>verb</i>	<i>noun</i>	<i>conj.</i>	<i>verb</i>	<i>adverb</i>

Word-level Classification

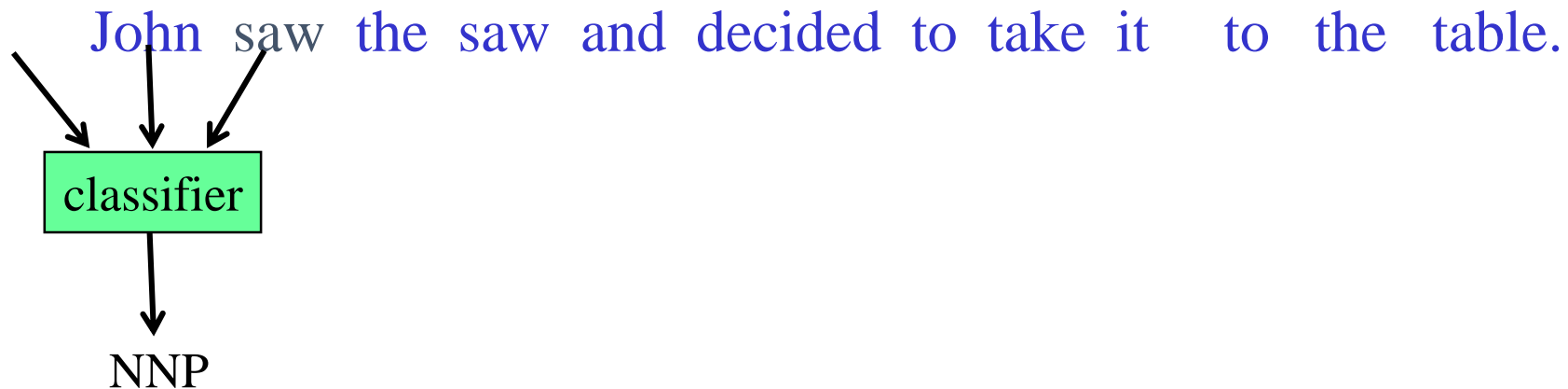
- We can do pretty well by classifying each word on its own
- Features:
 - Lowercase / uppercase
 - Prefixes / suffixes ('-ed' → verb, '-ly' → adverb, 'un-' → adjective)
 - Non-letter characters (periods → acronyms, only numbers → quantifier)

Bill	saw	that	man	yesterday
name	verb (past)	det.	noun	adverb
<i>verb</i>	<i>verb</i>	<i>det.</i>	<i>verb</i>	<i>adverb</i>
<i>verb</i>	<i>noun</i>	<i>conj.</i>	<i>verb</i>	<i>adverb</i>

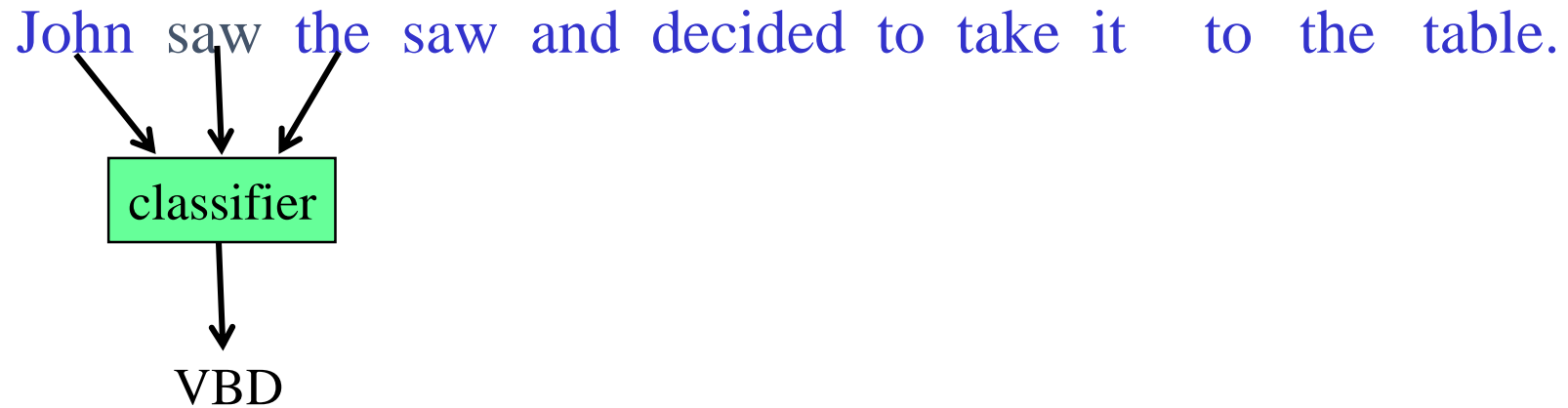
Not necessarily a bad option.
Think of: “find Mary before
coming back”

Sequence Labeling as Classification

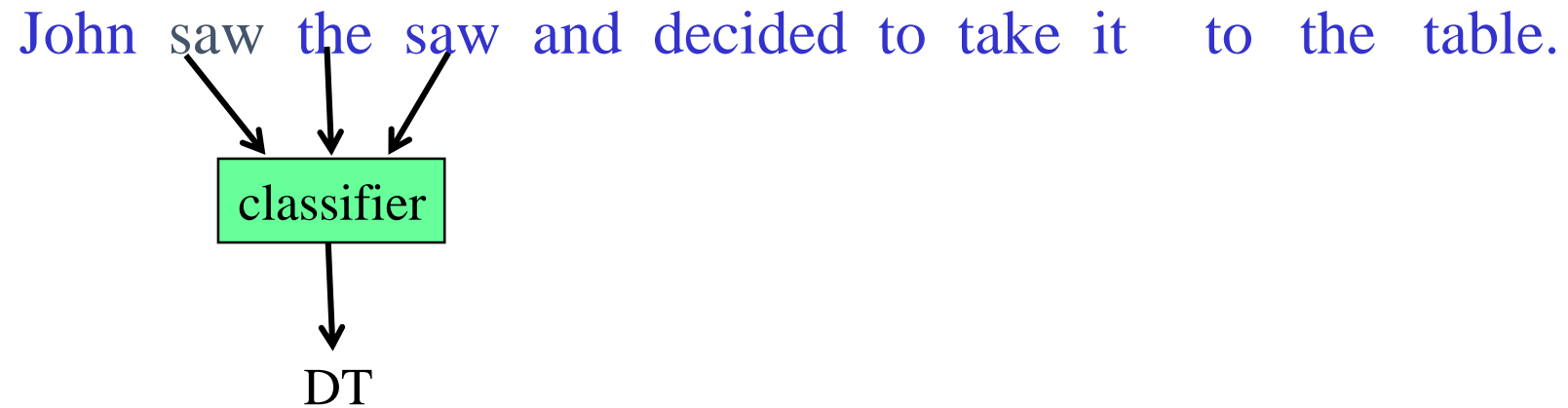
- **Option 1:** classify each token independently but use as input features, information about the surrounding tokens (sliding window)



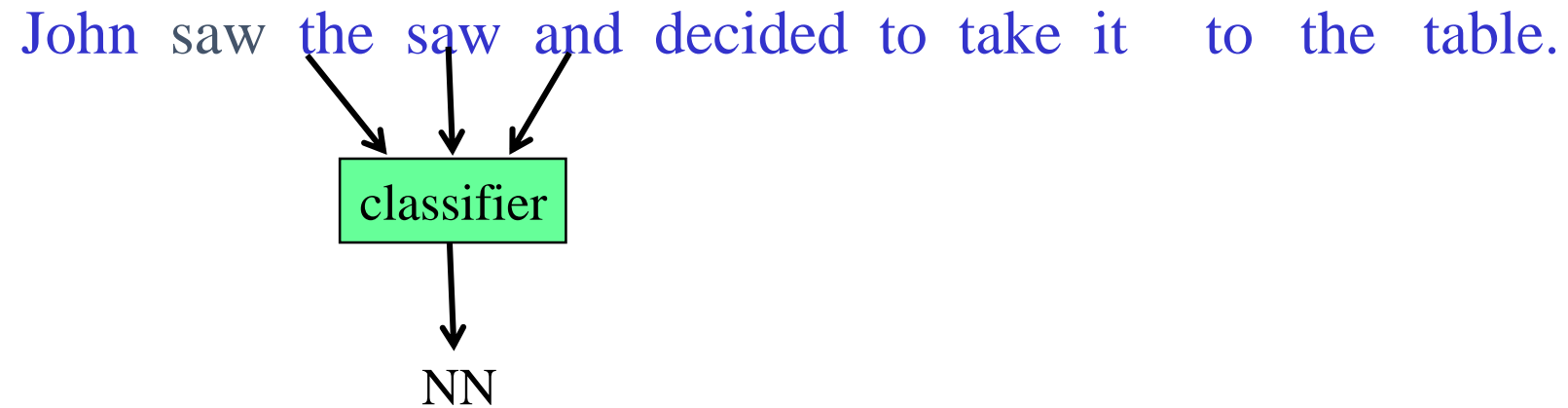
Sequence Labeling as Classification



Sequence Labeling as Classification

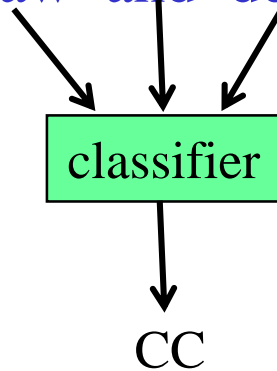


Sequence Labeling as Classification



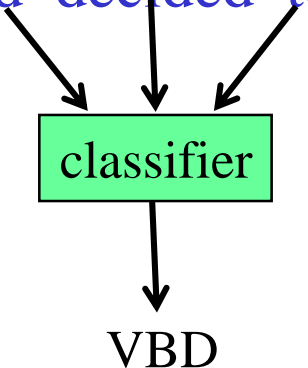
Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



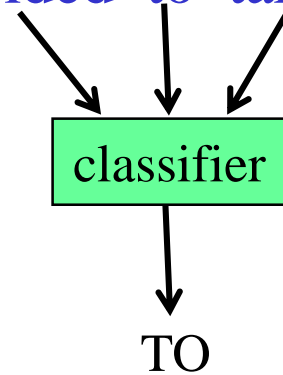
```
graph TD; A[John] --> C[classifier]; B[saw] --> C; D[the] --> C; E[saw] --> C; F[and] --> C; G[decided] --> C; H[to] --> C; I[take] --> C; J[it] --> C; K[to] --> C; L[the] --> C; M[table.] --> C; C --> VBD[VBD];
```

classifier

VBD

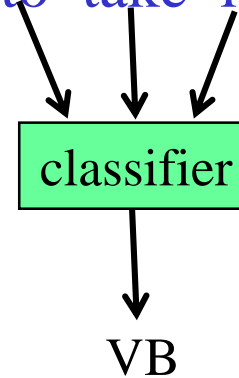
Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



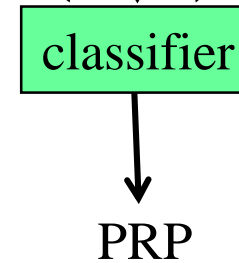
Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



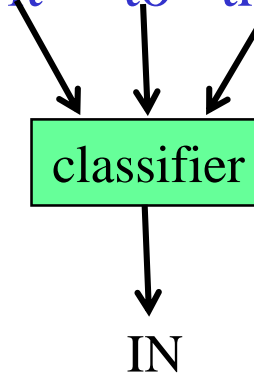
Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



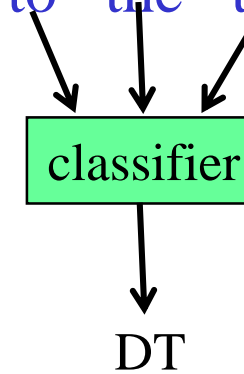
Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



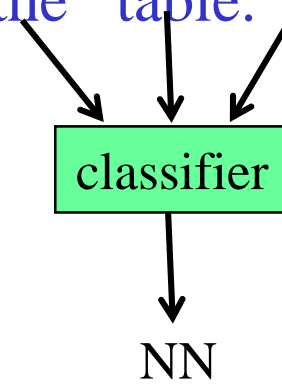
Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

John saw the saw and decided to take it to the table.



```
graph TD; A[the] --> C[classifier]; B[table.] --> C; D[to] --> C; C --> E[NN]
```

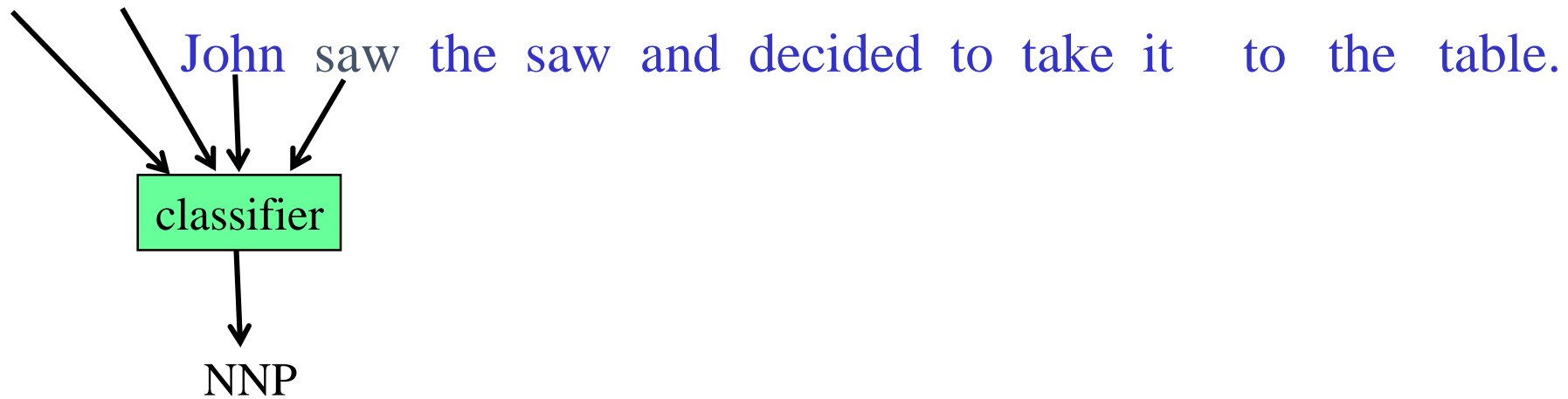
classifier

NN

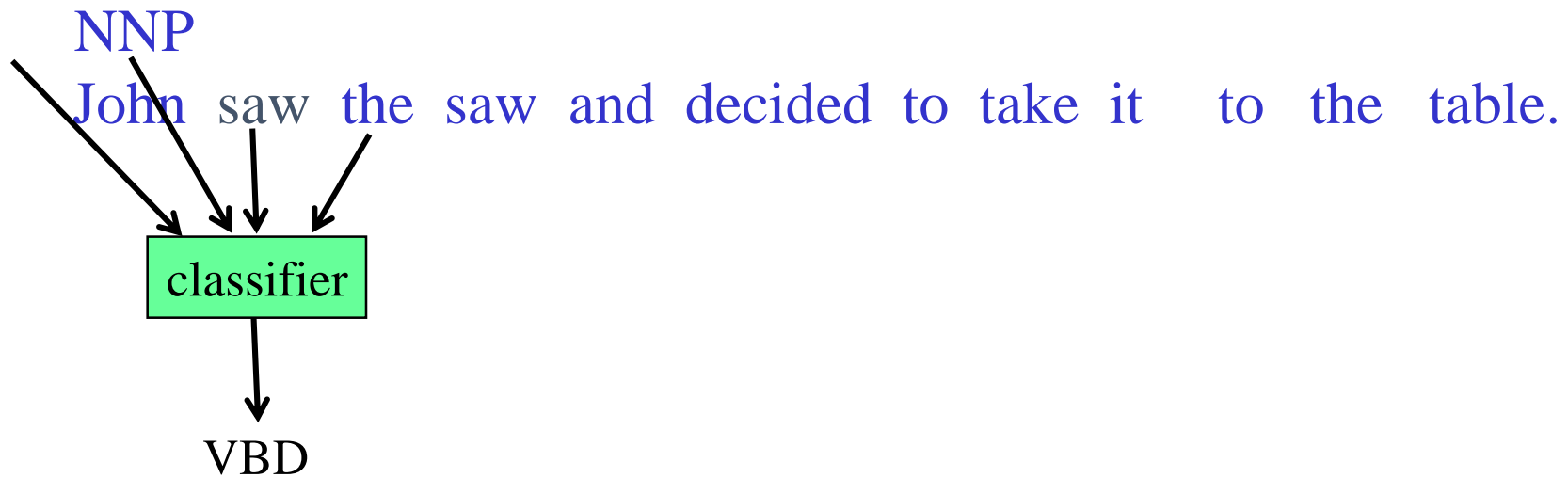
Sequence Labeling as Classification using Outputs as Inputs

- Better input features are usually the **categories** of the surrounding tokens, but these are not available yet
- Can use category of either the preceding or succeeding tokens by going forward or backward and using previous output

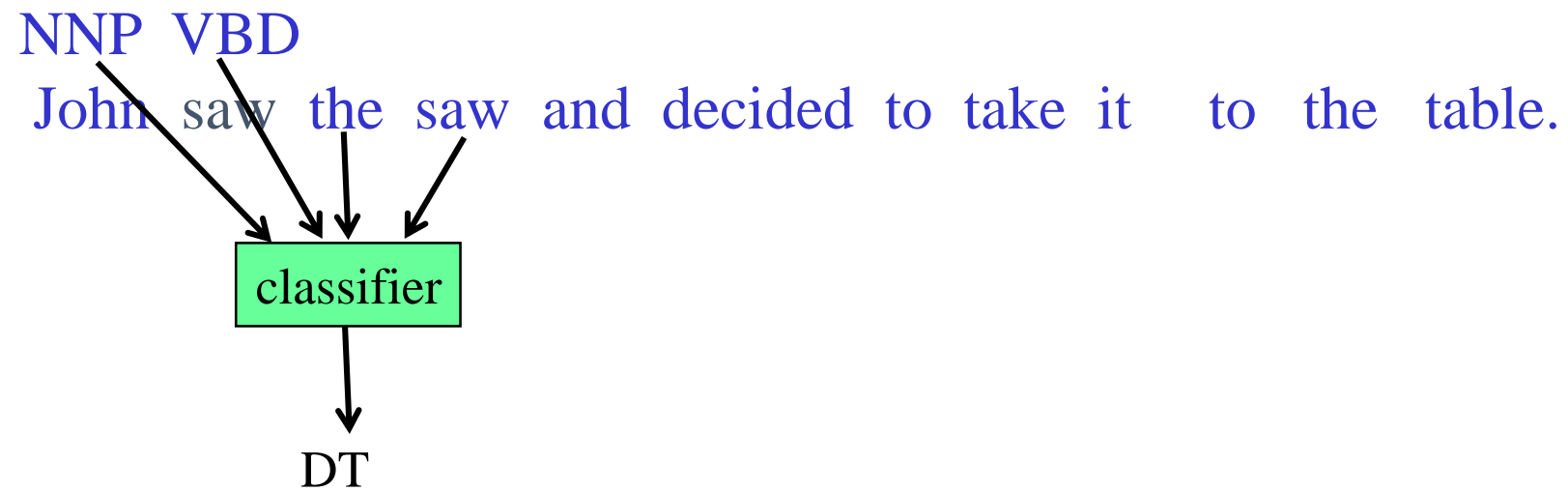
Forward Classification



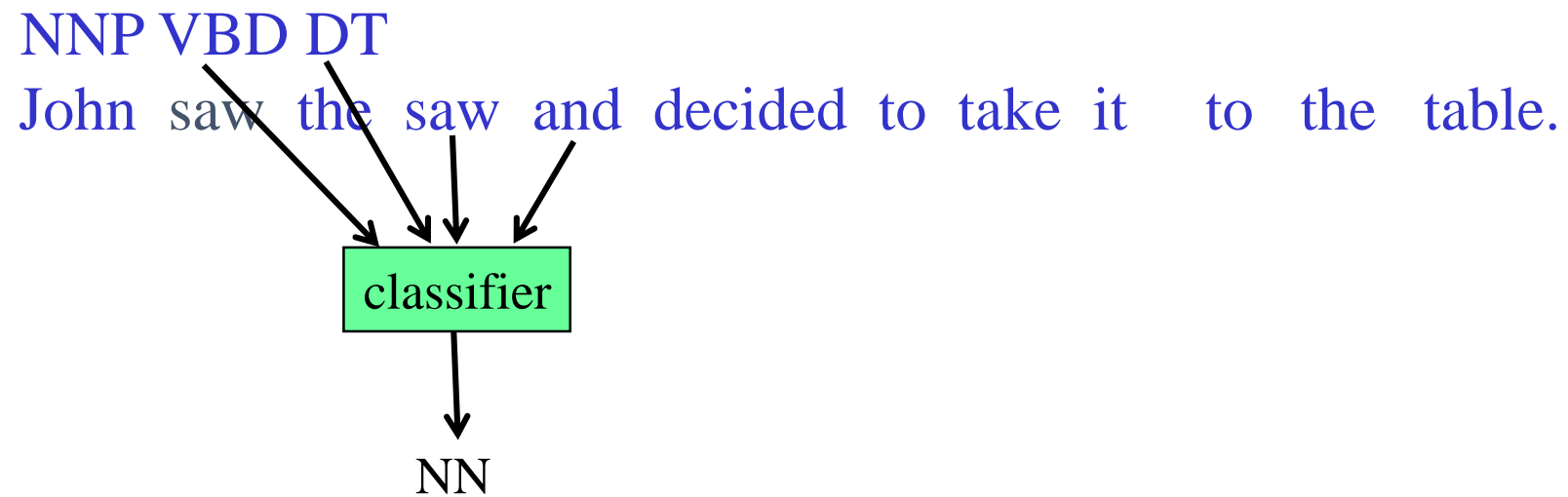
Forward Classification



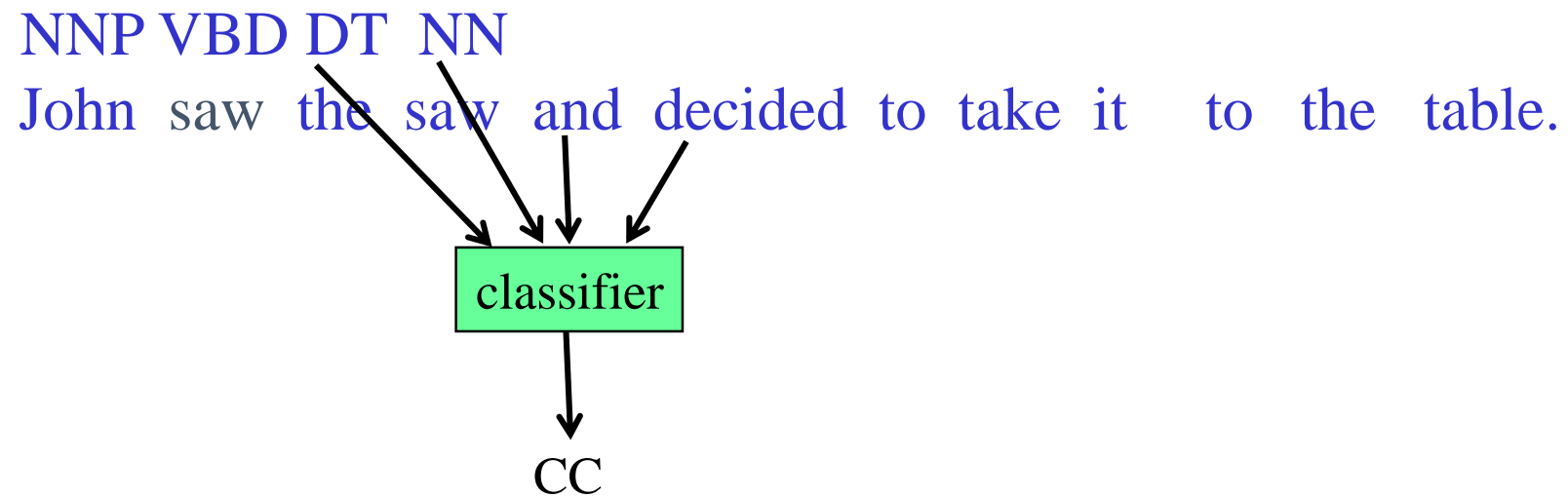
Forward Classification



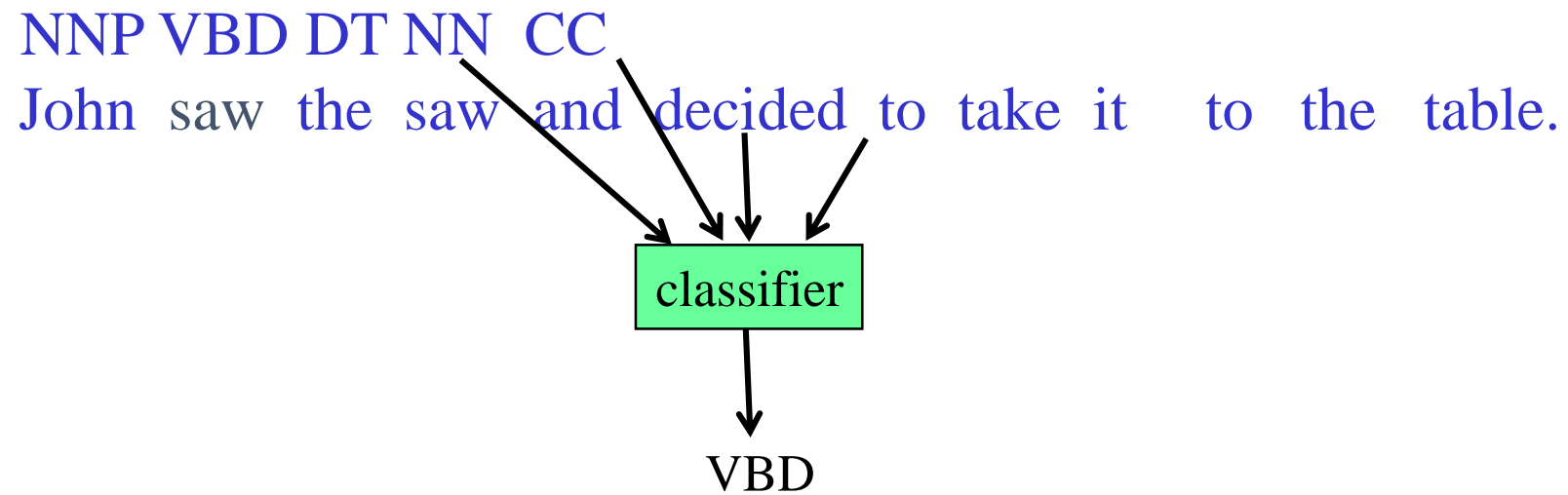
Forward Classification



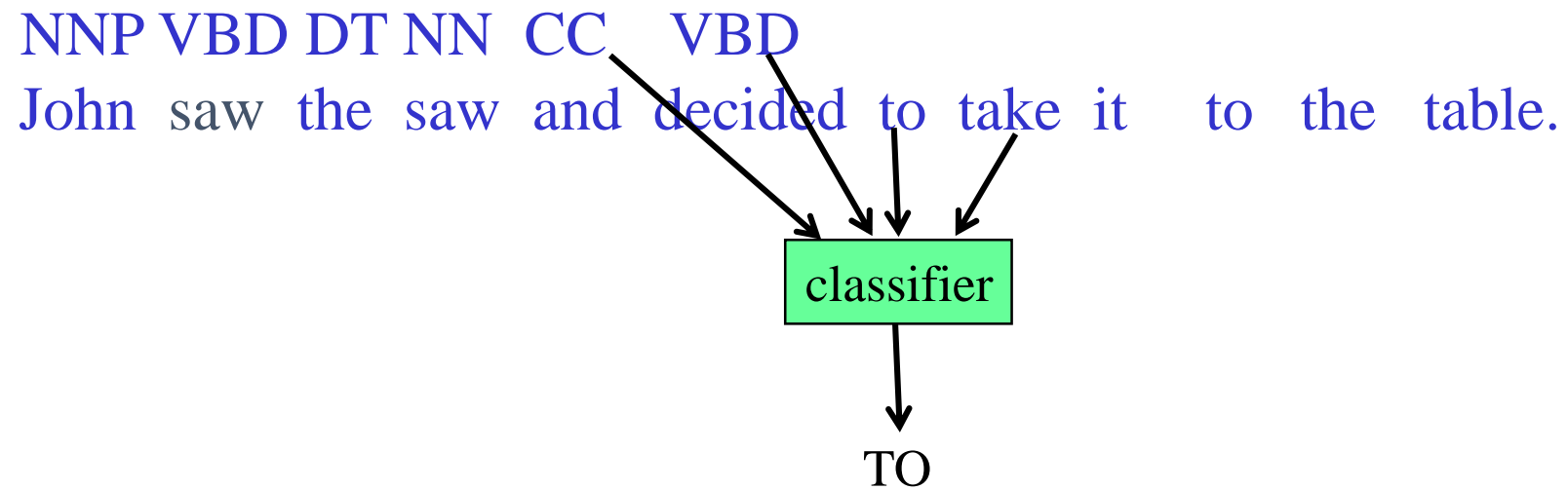
Forward Classification



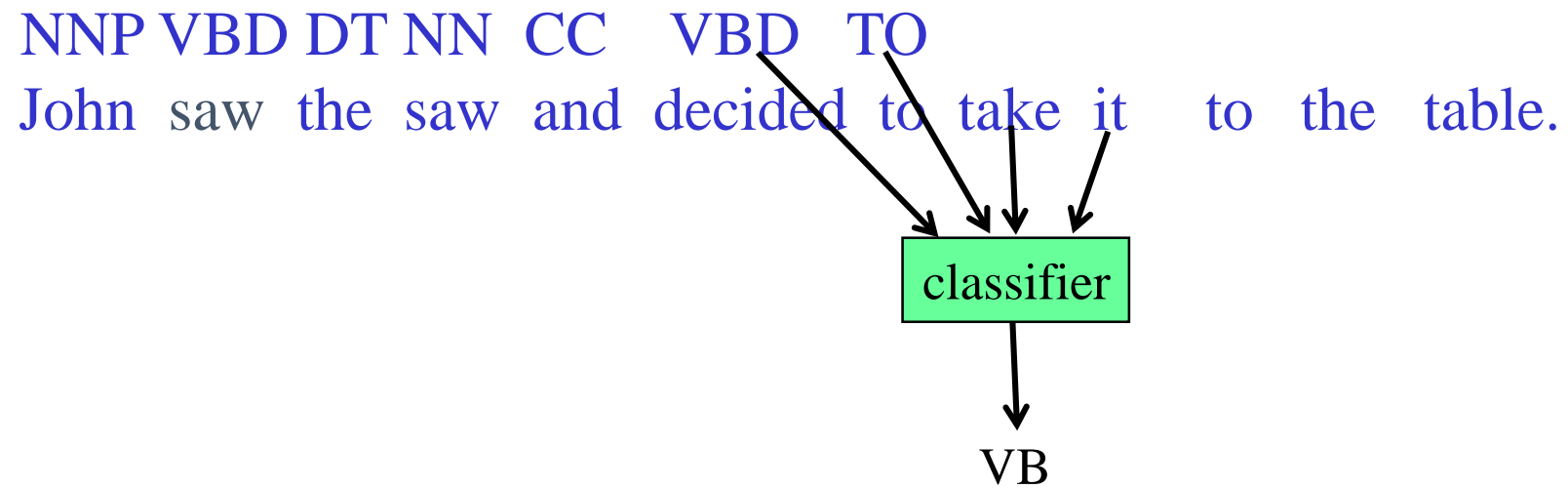
Forward Classification



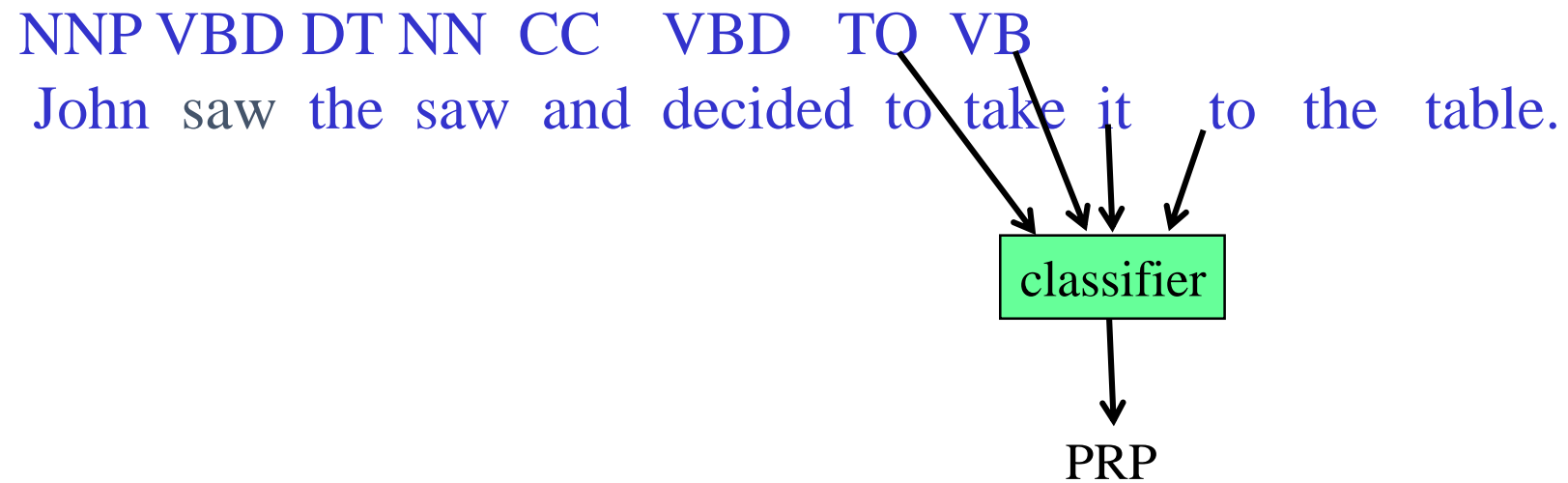
Forward Classification



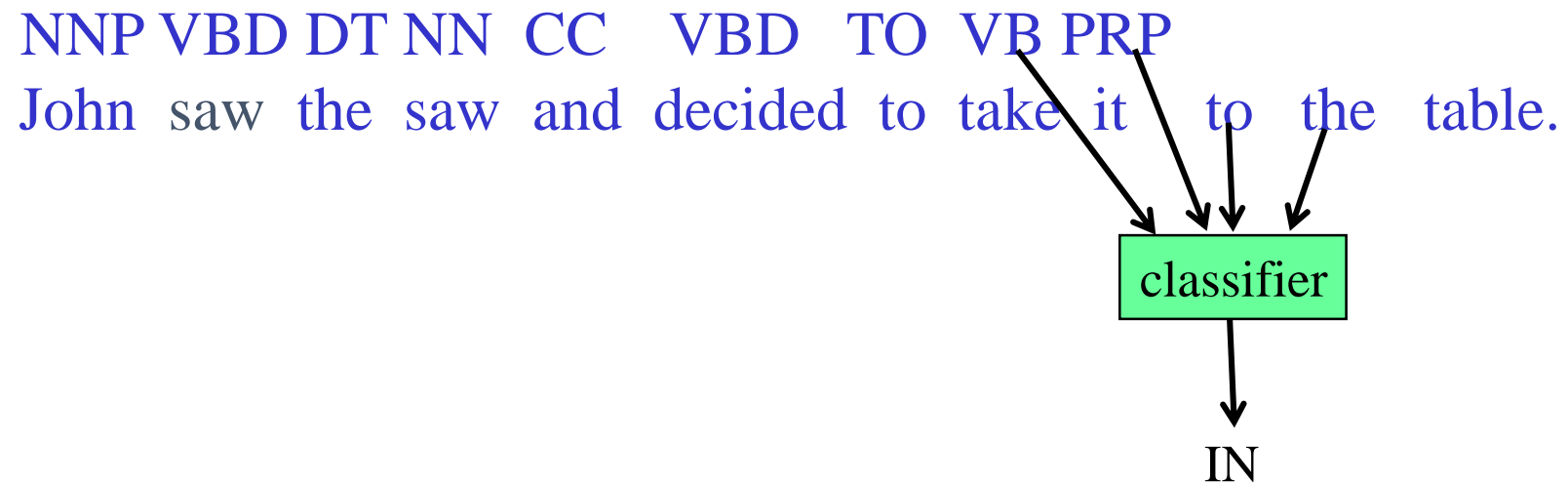
Forward Classification



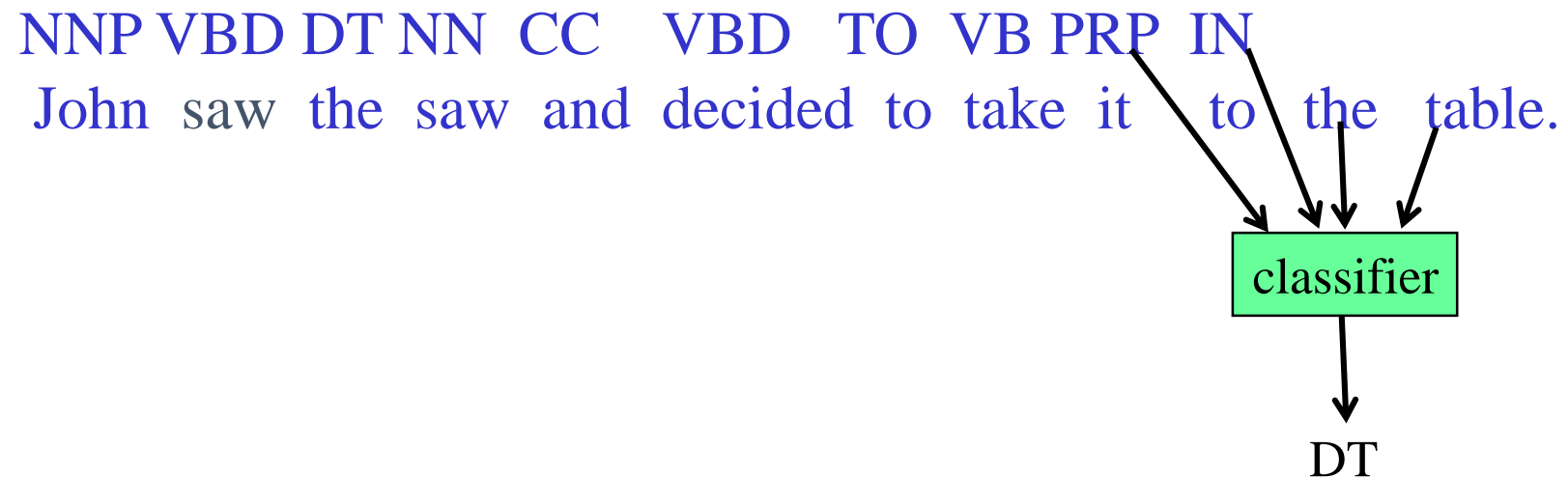
Forward Classification



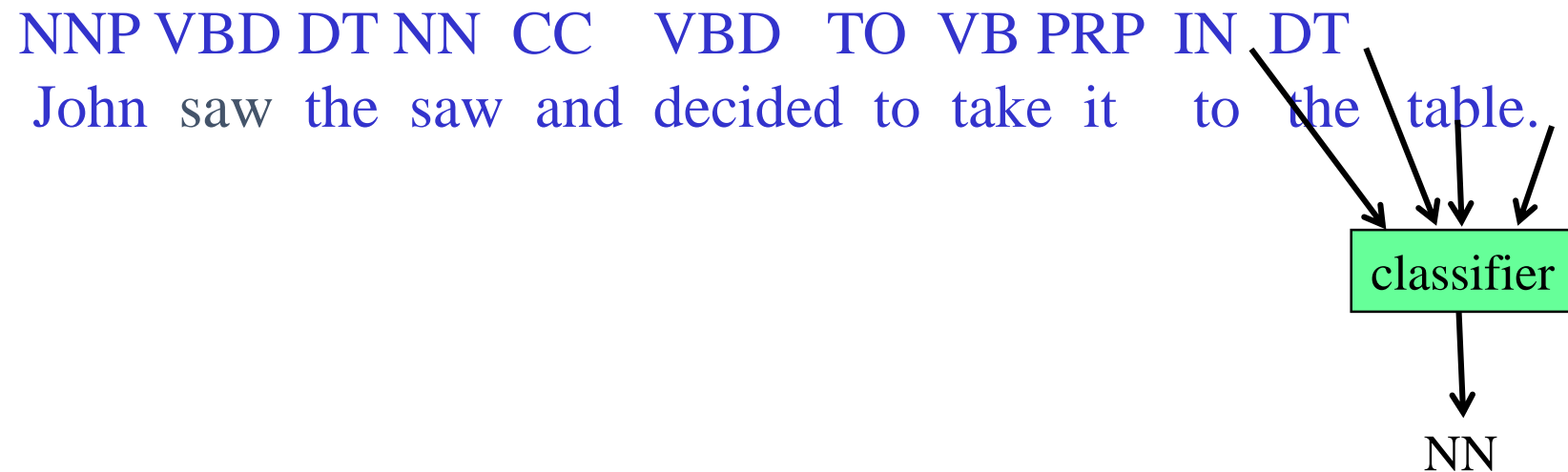
Forward Classification



Forward Classification

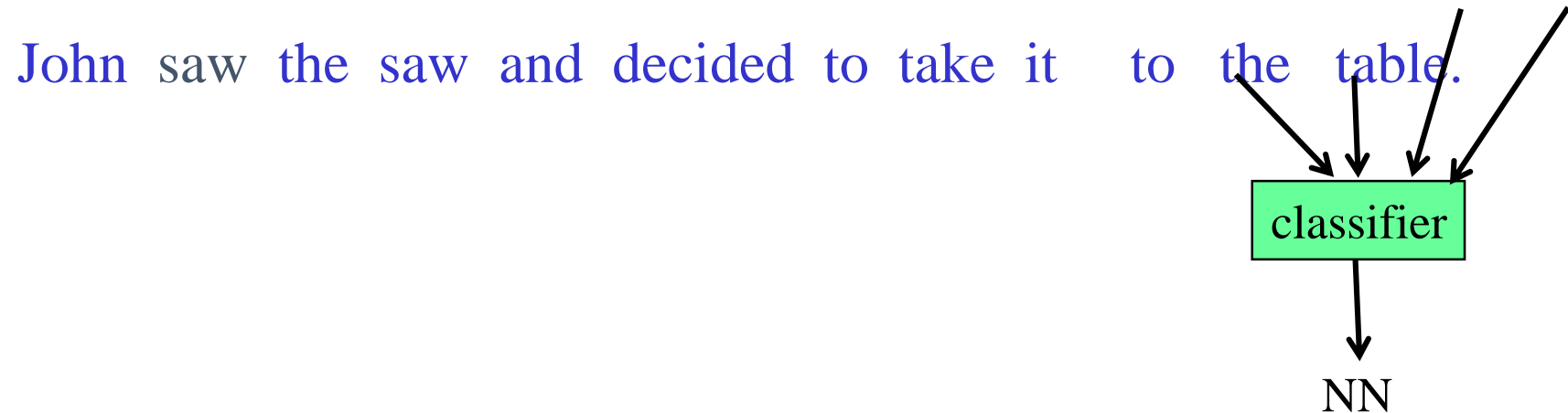


Forward Classification

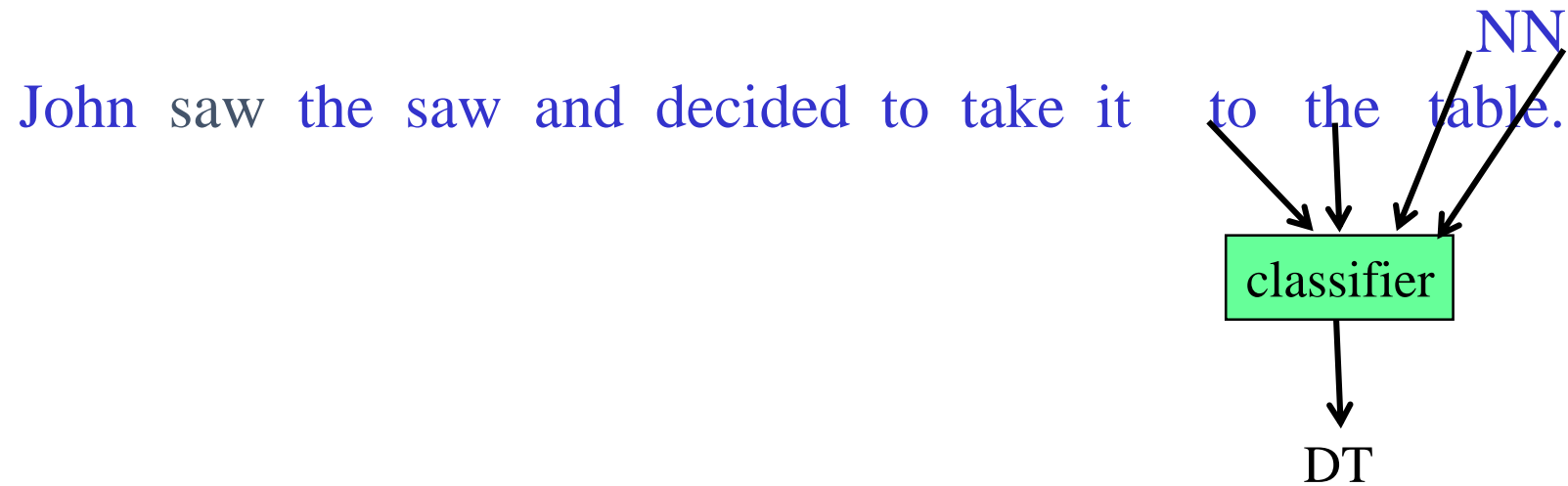


Backward Classification

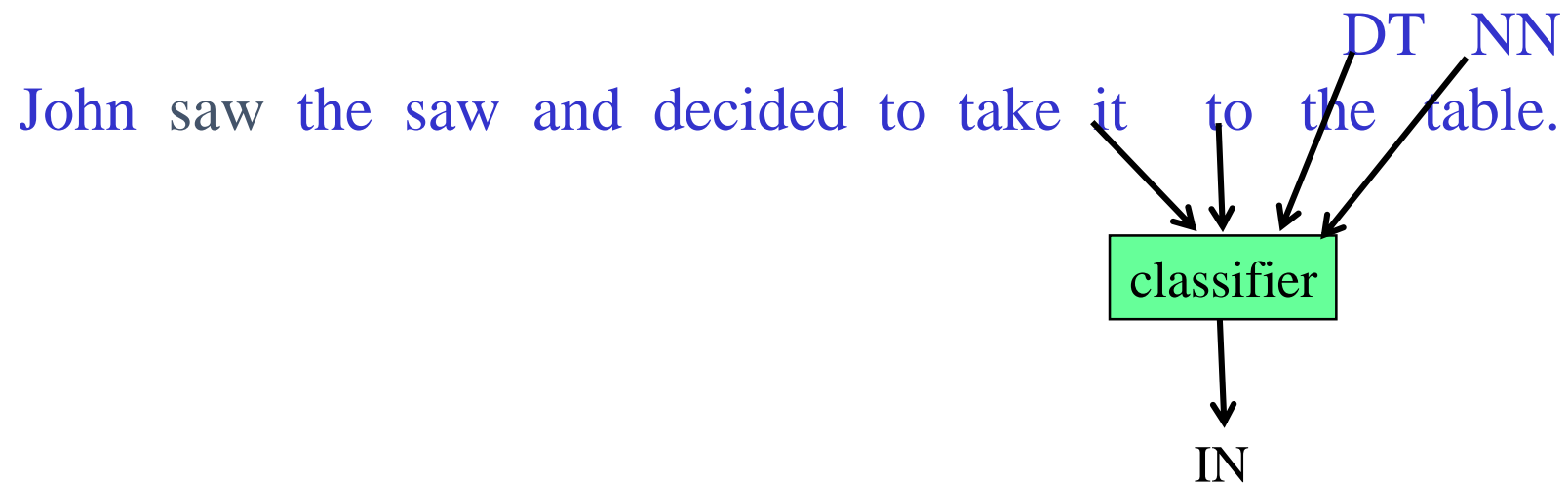
- Disambiguating “to” in this case would be easier backward



Backward Classification



Backward Classification

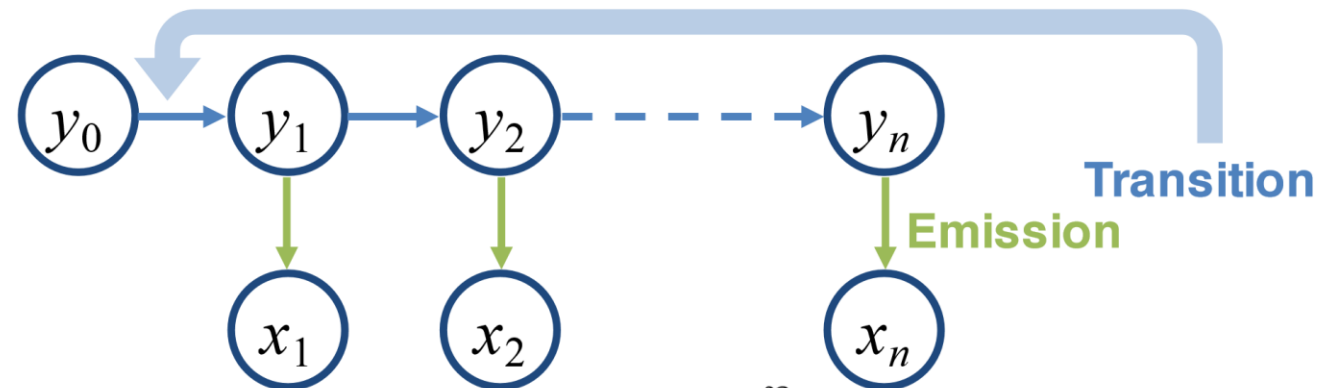


Problems with Sequence Labeling as Classification

- Not easy to integrate information from category of tokens on both sides
- Difficult to propagate uncertainty between decisions and “collectively” determine the most likely joint assignment of categories to all of the tokens in a sequence

Probabilistic Sequence Models

- Probabilistic sequence models allow integrating uncertainty over multiple, interdependent classifications and collectively determine the most likely global assignment
- Classic solution: Hidden Markov Models



$$p(x_1 \dots x_n, y_1 \dots y_n) = q(STOP|y_n) \prod_{i=1}^n q(y_i|y_{i-1})e(x_i|y_i)$$

Markov Models

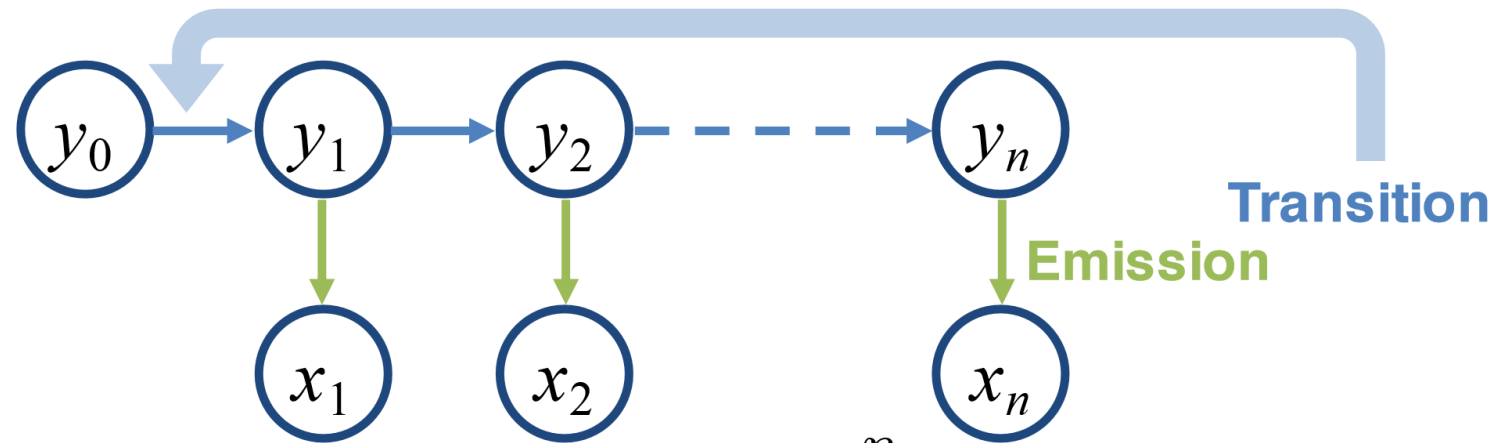
- Recall: Markov models assume the probability of a sequence can be given as a product of the *transition probabilities*

$$\Pr(x_1 x_2 \dots x_n) = \prod_i \Pr(x_i | x_{i-1})$$

- For brevity of notation, we sometimes assume that x_0 is a fixed START symbol
- Equivalent to a finite state machine with probabilistic state transitions

Hidden Markov Model

- In a hidden Markov model there are two sequences. One is a Markov chain (the y_i 's) and one is *emitted* from the Markov chain (x_i 's)
- A bi-gram (1st order Markov model) is:



$$p(x_1 \dots x_n, y_1 \dots y_n) = q(STOP|y_n) \prod_{i=1}^n q(y_i|y_{i-1})e(x_i|y_i)$$

Hidden Markov Model

- Just like with Markov language models, HMM can be of a higher-order
- For example, a 2nd order (tri-gram) HMM is

Hidden Markov Model

- Two important modeling assumptions:
 - The word (emitted value) x_i is independent of the rest of the variables given y_i
 - The POS sequence is Markovian, so a POS tag is independent of past tags given the previous tag
- POS tags don't meet either assumption
 - POS tags have dependencies that cannot be bounded in distance
 - The verb *give* usually takes two arguments, where the first can be arbitrarily long
 - Words are dependent on other words in the sentence, even given their POS
 - If you heard the verb *elapsed*, then it is likely the word *time*, *minutes*, *seconds* or *hours* also appeared in the sentence
- **Still, HMMs are a practical option**

Hidden Markov Models (HMMs)

- ▶ We have an input sentence $x = x_1, x_2, \dots, x_n$
(x_i is the i 'th word in the sentence)
- ▶ We have a tag sequence $y = y_1, y_2, \dots, y_n$
(y_i is the i 'th tag in the sentence)
- ▶ We'll use an HMM to define

$$p(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n)$$

for any sentence $x_1 \dots x_n$ and tag sequence $y_1 \dots y_n$ of the same length.

- ▶ Then the most likely tag sequence for x is

$$\arg \max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1, y_2, \dots, y_n)$$

Trigram HMM

For any sentence $x_1 \dots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \dots n$, and any tag sequence $y_1 \dots y_{n+1}$ where $y_i \in \mathcal{S}$ for $i = 1 \dots n$, and $y_{n+1} = \text{STOP}$, the joint probability of the sentence and tag sequence is

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i | y_i)$$

where we have assumed that $x_0 = x_{-1} = *$.

Trigram HMM

For any sentence $x_1 \dots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \dots n$, and any tag sequence $y_1 \dots y_{n+1}$ where $y_i \in \mathcal{S}$ for $i = 1 \dots n$, and $y_{n+1} = \text{STOP}$, the joint probability of the sentence and tag sequence is

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i | y_i)$$

where we have assumed that $x_0 = x_{-1} = *$.

Parameters of the model:

- ▶ $q(s|u, v)$ for any $s \in \mathcal{S} \cup \{\text{STOP}\}$, $u, v \in \mathcal{S} \cup \{*\}$
- ▶ $e(x|s)$ for any $s \in \mathcal{S}$, $x \in \mathcal{V}$

Trigram HMM

An Example

If we have $n = 3$, $x_1 \dots x_3$ equal to the sentence *the dog laughs*, and $y_1 \dots y_4$ equal to the tag sequence D N V STOP, then

$$p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

= ?

Trigram HMM

An Example

If we have $n = 3$, $x_1 \dots x_3$ equal to the sentence *the dog laughs*, and $y_1 \dots y_4$ equal to the tag sequence D N V STOP, then

$$\begin{aligned} & p(x_1 \dots x_n, y_1 \dots y_{n+1}) \\ = & q(D|*, *) \times q(N|*, D) \times q(V|D, N) \times q(\text{STOP}|N, V) \\ & \times e(\text{the}|D) \times e(\text{dog}|N) \times e(\text{laughs}|V) \end{aligned}$$

- ▶ STOP is a special tag that terminates the sequence
- ▶ We take $y_0 = y_{-1} = *$, where $*$ is a special “padding” symbol

MLE in Trigram HMMs

- Given training data in the form of sequences of words and corresponding sequences of labels, we can estimate the transition (q) and emission (e) probability by relative frequencies:

$$\hat{q}(y_3|y_2, y_1) = \frac{\#(y_1, y_2, y_3)}{\sum_y \#(y_1, y_2, y)}$$

$$\hat{e}(x_1|y_1) = \frac{\#(x_1, y_1)}{\sum_x \#(x, y_1)}$$

- Smoothing techniques are applicable

Trigram HMM

Smoothed Estimation

The transition parameters can be smoothed just like n-gram models

$$q(V_t | D_t, J_t) = \lambda_1 \times \frac{\text{Count}(D_t, J_t, V_t)}{\text{Count}(D_t, J_t)} \\ + \lambda_2 \times \frac{\text{Count}(J_t, V_t)}{\text{Count}(J_t)} \\ + \lambda_3 \times \frac{\text{Count}(V_t)}{\text{Count}()}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1, \quad \text{and for all } i, \lambda_i \geq 0$$

For emission probabilities we usually use pseudowords

$$e(\text{base} | V_t) = \frac{\text{Count}(V_t, \text{base})}{\text{Count}(V_t)}$$

HMM: Smoothing with Pseudowords

Pseudowords use our existing information about the task to categorize words that appeared very few times, or not at all, in the training data

A common method is as follows:

- ▶ **Step 1:** Split vocabulary into two sets

Frequent words = words occurring ≥ 5 times in training

Low frequency words = all other words

- ▶ **Step 2:** Map low frequency words into a small, finite set, depending on prefixes, suffixes etc.

HMM: Smoothing with Pseudowords

[[Bikel et. al 1999](#)] (named-entity recognition)

Word class	Example	Intuition
twoDigitNum	90	Two digit year
fourDigitNum	1990	Four digit year
containsDigitAndAlpha	A8956-67	Product code
containsDigitAndDash	09-96	Date
containsDigitAndSlash	11/9/89	Date
containsDigitAndComma	23,000.00	Monetary amount
containsDigitAndPeriod	1.00	Monetary amount, percentage
othernum	456789	Other number
allCaps	BBN	Organization
capPeriod	M.	Person name initial
firstWord	first word of sentence	no useful capitalization information
initCap	Sally	Capitalized word
lowercase	can	Uncapitalized word
other	,	Punctuation marks, all other words

HMM: Smoothing with Pseudowords

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA
topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA
CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA
results/NA ./NA



firstword/NA soared/NA at/NA initCap/SC Co./CC ,/NA easily/NA
lowercase/NA forecasts/NA on/NA initCap/SL Street/CL ,/NA as/NA
their/NA CEO/NA Alan/SP initCap/CP announced/NA first/NA
quarter/NA results/NA ./NA

NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location

HMM: Inference

The Viterbi Algorithm

Problem: for an input $x_1 \dots x_n$, find

$$\arg \max_{y_1 \dots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

where the $\arg \max$ is taken over all sequences $y_1 \dots y_{n+1}$ such that $y_i \in \mathcal{S}$ for $i = 1 \dots n$, and $y_{n+1} = \text{STOP}$.

We assume that p again takes the form

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i | y_i)$$

Recall that we have assumed in this definition that $y_0 = y_{-1} = *$, and $y_{n+1} = \text{STOP}$.

HMM: Inference

Brute Force Search is Hopelessly Inefficient

Problem: for an input $x_1 \dots x_n$, find

$$\arg \max_{y_1 \dots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

where the $\arg \max$ is taken over all sequences $y_1 \dots y_{n+1}$ such that $y_i \in \mathcal{S}$ for $i = 1 \dots n$, and $y_{n+1} = \text{STOP}$.

HMM: Inference

The Viterbi Algorithm

- ▶ Define n to be the length of the sentence
- ▶ Define S_k for $k = -1 \dots n$ to be the set of possible tags at position k :

$$S_{-1} = S_0 = \{*\}$$
$$S_k = S \quad \text{for } k \in \{1 \dots n\}$$

- ▶ Define

$$r(y_{-1}, y_0, y_1, \dots, y_k) = \prod_{i=1}^k q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^k e(x_i | y_i)$$

- ▶ Define a dynamic programming table

$$\pi(k, u, v) = \text{maximum probability of a tag sequence} \\ \text{ending in tags } u, v \text{ at position } k$$

that is,

$$\pi(k, u, v) = \max_{\langle y_{-1}, y_0, y_1, \dots, y_k \rangle : y_{k-1}=u, y_k=v} r(y_{-1}, y_0, y_1 \dots y_k)$$

HMM: Inference

A Recursive Definition

Base case:

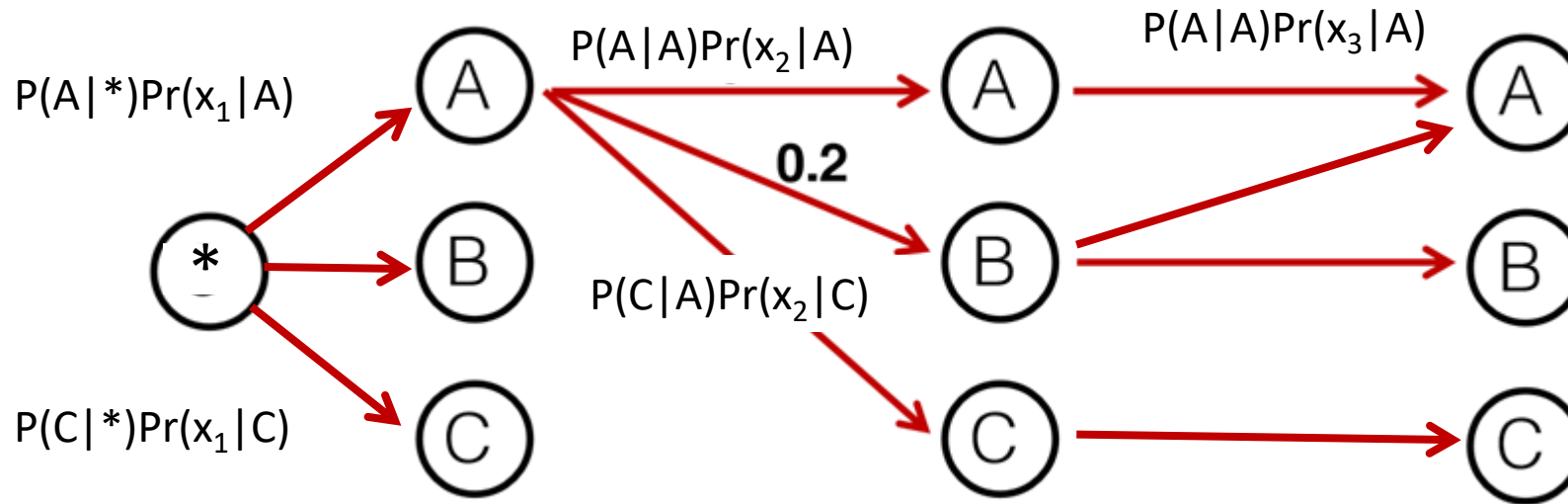
$$\pi(0, *, *) = 1$$

Recursive definition:

For any $k \in \{1 \dots n\}$, for any $u \in \mathcal{S}_{k-1}$ and $v \in \mathcal{S}_k$:

$$\pi(k, u, v) = \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

HMM: Graphic Presentation of Viterbi Alg.



- Finding the most likely assignment for x_1, \dots, x_n that ends with a certain state s is equivalent to finding the maximum weighted path from $*$ to the state s in the n -th layer
 - The weight of a path is here the product of the weights of its edges
- a dynamic programming approach solves the inference problem

HMM: Inference with the Viterbi Algorithm

The Viterbi Algorithm

Input: a sentence $x_1 \dots x_n$, parameters $q(s|u, v)$ and $e(x|s)$.

Initialization: Set $\pi(0, *, *) = 1$

Definition: $\mathcal{S}_{-1} = \mathcal{S}_0 = \{*\}$, $\mathcal{S}_k = \mathcal{S}$ for $k \in \{1 \dots n\}$

Algorithm:

- ▶ For $k = 1 \dots n$,

- ▶ For $u \in \mathcal{S}_{k-1}$, $v \in \mathcal{S}_k$,

$$\pi(k, u, v) = \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

- ▶ **Return** $\max_{u \in \mathcal{S}_{n-1}, v \in \mathcal{S}_n} (\pi(n, u, v) \times q(\text{STOP}|u, v))$

HMM: Inference with the Viterbi Algorithm

The Viterbi Algorithm with Backpointers

Input: a sentence $x_1 \dots x_n$, parameters $q(s|u, v)$ and $e(x|s)$.

Initialization: Set $\pi(0, *, *) = 1$

Definition: $\mathcal{S}_{-1} = \mathcal{S}_0 = \{*\}$, $\mathcal{S}_k = \mathcal{S}$ for $k \in \{1 \dots n\}$

Algorithm:

- ▶ For $k = 1 \dots n$,
 - ▶ For $u \in \mathcal{S}_{k-1}$, $v \in \mathcal{S}_k$,
$$\pi(k, u, v) = \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$
$$bp(k, u, v) = \arg \max_{w \in \mathcal{S}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$
- ▶ Set $(y_{n-1}, y_n) = \arg \max_{(u,v)} (\pi(n, u, v) \times q(\text{STOP}|u, v))$
- ▶ For $k = (n-2) \dots 1$, $y_k = bp(k+2, y_{k+1}, y_{k+2})$
- ▶ **Return** the tag sequence $y_1 \dots y_n$

HMM: Inference with the Viterbi Algorithm

The Viterbi Algorithm: Running Time

- ▶ $O(n|\mathcal{S}|^3)$ time to calculate $q(s|u, v) \times e(x_k|s)$ for all k, s, u, v .
- ▶ $n|\mathcal{S}|^2$ entries in π to be filled in.
- ▶ $O(|\mathcal{S}|)$ time to fill in one entry

HMM: Pros and Cons

- Pros:
 - A simple and widely used model with competitive performance in many sequence labeling tasks
 - Very simple to train
 - Inference is efficient
- Cons:
 - Strong independence assumption
 - Inability to use features to model the emission distribution