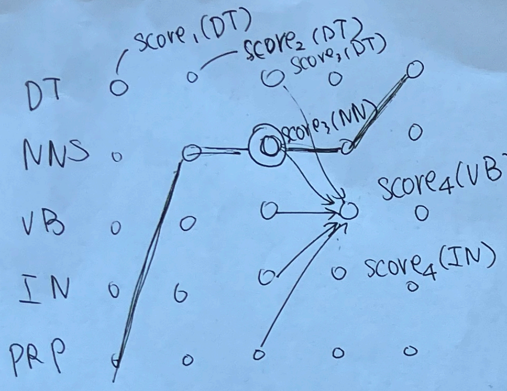


$$\mathrm{score}_4(VB) = \max_{y_3} \begin{cases} P(VB|DT)\,P(\text{and}|VB)\cdot \mathrm{score}_3(DT) \\ P(VB|NNS)\,P(\text{and}|VB)\cdot \mathrm{score}_3(NNS) \\ P(VB|VB) \ldots \\ P(VB|IN) \ldots \\ P(PRP|NNS)\,P(\text{and}|VB)\cdot \mathrm{score}_3(PRP) \end{cases}$$

$P(y_4|y_3)P(x_4|y_4)\cdot$

$$\mathrm{score}_4(IN) = \max_{y_3} \begin{cases} \ldots \mathrm{score}_3(DT) \\ \ldots \mathrm{score}_3(NNS) \\ \ldots \\ \ldots \mathrm{score}_3(PRP) \end{cases}$$

I love cats and dogs .

x_1 x_2 x_3 x_4 x_5

"cats" NNS

$P(y_3 = NNS | \mathbf{x})$

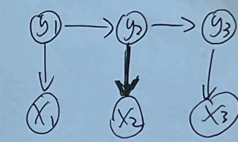
$\mathbf{y}$ = PRP NNS NNS NNS DT

$\mathbf{x}$ = I love cats and dogs

HMM

$n = 3$

x_1	x_2	x_3
I	love	cats
PRP	VBP	NNS
y_1	y_2	y_3



. edges convey conditional independence

i.e. x_2 is conditionally independent of everything else give y_2

$x_2 \perp\!\!\!\perp y_1, x_1 \mid y_2$.

$P(\mathbf{x}, \mathbf{y})$

$= P(x_1, x_2, x_3, y_1, y_2, y_3)$

$= P(y_1)\cdot P(x_1|y_1)\cdot \underbrace{P(y_2|y_1, x_1)}_{= P(y_2|y_1)}\, \underbrace{P(x_2|y_1, x_1, y_2)}_{= P(x_2|y_2)}\, \underbrace{P(y_3|y_1, x_1, y_2, x_2)}_{= P(y_3|y_2)}$

$\underbrace{P(x_3|y_1, x_1, y_2, x_2, y_3)}_{= P(x_3|y_3)}$

Dependency Grammars.

syntactic structure = lexical items linked by binary asymmetrical relations called dependency

$\text{Head} \xrightarrow{\text{dependency type}} \text{Dependent}$

(modifier/object/compliment

Dynamic Programming

1, 1, 2, 3, 5, 8, 13, . . .

Fibonacci Numbers: $F_1 = F_2 = 1$; $F_n = F_{n-1} + F_{n-2}$

∘ naive algorithm

```
fib(n):
    if n ≤ 2, return 1
    else return fib(n-1) + fib(n-2).
```

$\Longrightarrow$ $T(0) = T(1) = 1$

$$T(n) = T(n-1) + T(n-2) + \underbrace{O(1)}_{\text{constant}} = c$$

$$\geq 2T(n-2) + O(1) = c$$

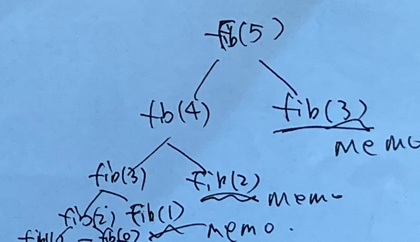
$$\geq 2(2T(n-4)+c)+c = 4T(n-4) + 3c$$

$$\vdots$$

$$\geq 2^k T(n-2k) + (2^k - 1)c$$

when $n - 2k = 0 \Longrightarrow k = n/2$

$$T(n) \sim 2^{n/2} \cdot T(0)$$



∘ dynamic programming

```
memo = {}
fib(n):
    if n in memo, return memo[n]
    else: if n ≤ 2 : f = 1
          else: f = fib(n-1) + fib(n-2)
          memo[n] = f
          return f.
```

$\underline{\qquad}$ free lookup.

$\Longrightarrow$ $T(n) = T(n-1) + O(1)$

$= O(n)$.