PRON Verb $z_3$ - - -
I like to eat - - -
PRON Verb - - -
She likes to work out - -
$z_1$ $z_2$ - - -
There is - - -

$\pi_0 = [\ \cdot\ ] \rightarrow \pi_0(\text{PRON})$

$= \frac{\text{Count (PRON in 1st places)}}{\#\text{ sentences}} = \frac{2}{3}$

# Natural Language Processing

## CSE 325/425

$A_0 = [\ \cdot\ ]$

$\pi, A, B$

Sihong Xie

$a^0_{\text{PRON}\rightarrow\text{Verb}} = \frac{\text{Count (PRON}\rightarrow\text{verb)}}{\text{Count (PRON)}} = \frac{2}{2} = 1$

Lecture 20:

- Project 2 discussion
- Neural CYK

arg of em( )

$A \leftarrow \mu A_L + (1-\mu) A_0$

$A_L = A_0$, $A_0 = A$ M-step

Not-so-close-example

$\begin{bmatrix} 3 & 2 \\ 3 & 6 \end{bmatrix} = 1 \times A_L = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + 2 \times A_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

# Neural CYK

2018 ~ 2019

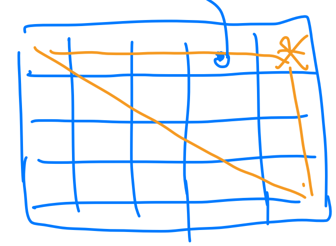
- Motivation: don't construct a CFG, but learn to predict a parse tree directly from a sentence. $P(T \mid S) = f(T; S)$
- Similar to CRF: no need to learn the A, B, and π matrices in HMM, but learn a parametric model to predict a POS-tag sequence from a sentence. $P(Q, O)$

$$P_1(Q \mid O) = \frac{1}{Z} \exp\Big\{ \sum_{t=1}^{T-1} \underbrace{\vec{w}^T \vec{f}(o_t, q_t, q_{t+1})}_{\text{can be replaced with NN output.}} \Big\}$$

  - Discriminative
- Similar to RNN/LSTM: use neural networks to combine rich information in a data-driven way
  - No hand-designed features.

# Neural CYK

$\delta_i(p,q)$ = score for (p,q) w i



- Recall: what we need to find the optimal parse tree in CYK.
  - the scores for each span ($p,q$) for each non-terminal $N^i$
  - the scores are computed based on a model (PCFG) and solutions to subproblems with a smaller span ($p,r$) and ($r+1,q$)

$$\psi_i(p,q) = \arg\max_{(j,k,r)} P(N^i \to N^j\ N^k)\delta_j(p,r)\delta_k(r+1,q)$$

$$\delta_i(p,q) = \max_{\substack{1\le j,k\le n\\ p\le r<q}} P(N^i \to N^j\ N^k)\delta_j(p,r)\delta_k(r+1,q)$$

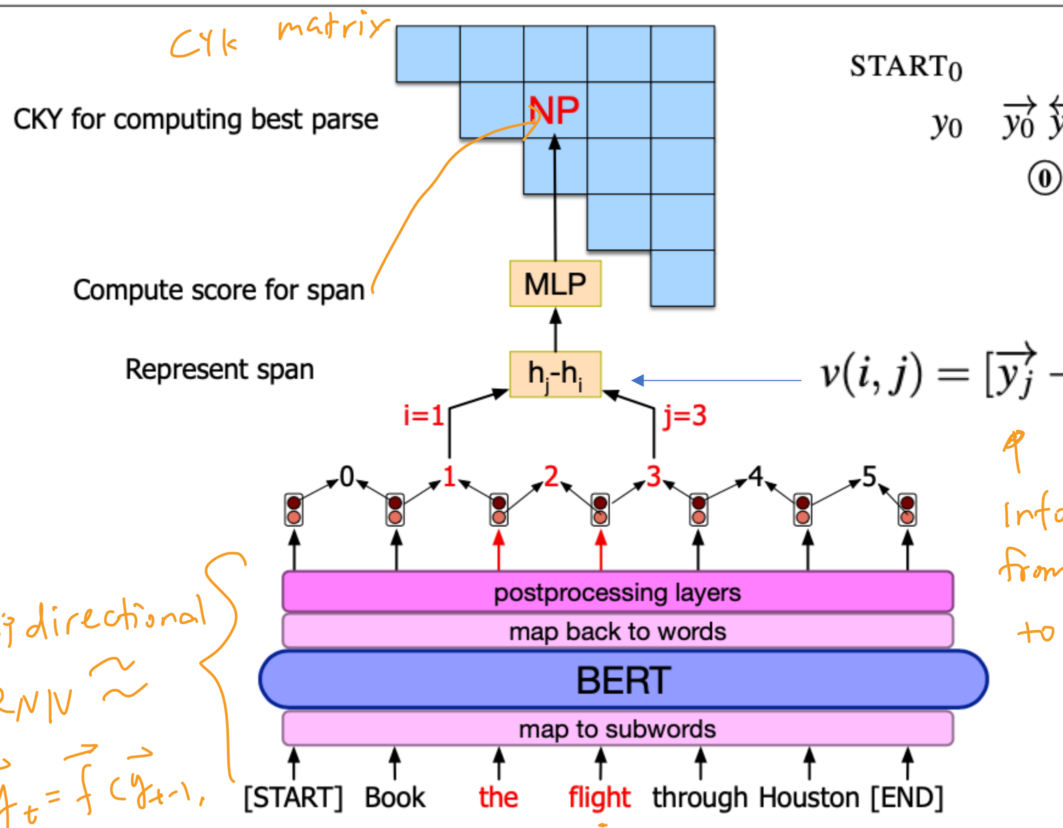
$$P(\hat{t}) = \delta_1(1,m)$$

  - Replace the probability $\Pr(N^i \to N^j N^k)$ with some scores predicted by a Neural network.

Prob CYK inside prob: $\beta_{NP}(i,j)$

# Neural CYK

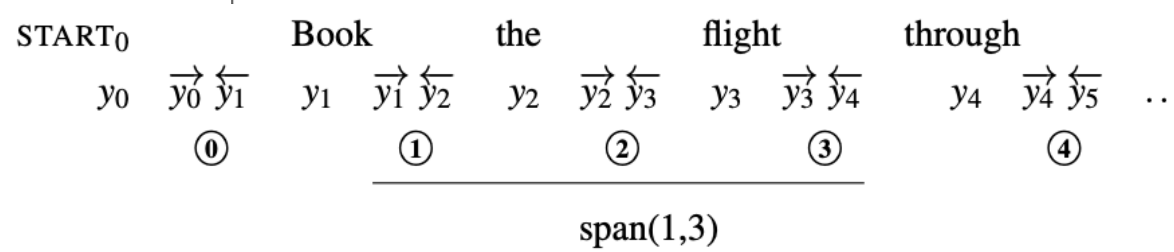
CYK matrix




Bidirectional RNN ~

$\vec{y}_t = f(\vec{y}_{t-1}, o_t)$

i j




$$v(i,j) = [\overrightarrow{y_j} - \overrightarrow{y_i}\ ;\ \overleftarrow{y_{j+1}} - \overleftarrow{y_{i+1}}]$$

from start to i

Info from START to j

What information is contained in the span (i, j), when read the sentence from left-to-right and right-to-left.

Score of the span (i, j):

$$s(i,j,\cdot) = \mathbf{W}_2\ \mathrm{ReLU}(\mathrm{LayerNorm}(\mathbf{W}_1 \mathbf{v}(i,j)))$$

We will talk about BERT towards the end of the course.

$\overleftarrow{y}_t = f(\overleftarrow{y}_{t+1}, o_t)$

Not Context free

due to Bidirectional RNN

# Neural CYK

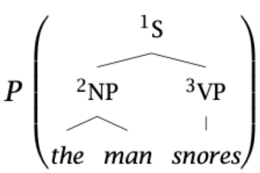
No CFG rules

- Finding the optimal tree
  - Assume that the scores of all spans are independent.
    - Related to the PCFG assumptions

A parse tree $T = \{\underbrace{(i_t, j_t, l_t)}_{\text{"clique" in CRF}} : t = 1, \ldots, |T|\}$

A optimal tree $\hat{T} = \underset{T}{\operatorname{argmax}}\, s(T)$

Score of a tree $s(T) = \sum_{(i,j,l) \in T} s(i,j,l)$

$P$ 


$$
\begin{aligned}
&= P({}^1\mathrm{S}_{13} \to {}^2\mathrm{NP}_{12}\ {}^3\mathrm{VP}_{33}, {}^2\mathrm{NP}_{12} \to \textit{the}_1\ \textit{man}_2, {}^3\mathrm{VP}_{33} \to \textit{snores}_3) \\
&= P({}^1\mathrm{S}_{13} \to {}^2\mathrm{NP}_{12}\ {}^3\mathrm{VP}_{33}) P({}^2\mathrm{NP}_{12} \to \textit{the}_1\ \textit{man}_2 | {}^1\mathrm{S}_{13} \to {}^2\mathrm{NP}_{12}\ {}^3\mathrm{VP}_{33}) \\
&\quad P({}^3\mathrm{VP}_{33} \to \textit{snores}_3 | {}^1\mathrm{S}_{13} \to {}^2\mathrm{NP}_{12}\ {}^3\mathrm{VP}_{33}, {}^2\mathrm{NP}_{12} \to \textit{the}_1\ \textit{man}_2) \\
&= P({}^1\mathrm{S}_{13} \to {}^2\mathrm{NP}_{12}\ {}^3\mathrm{VP}_{33}) P({}^2\mathrm{NP}_{12} \to \textit{the}_1\ \textit{man}_2) P({}^3\mathrm{VP}_{33} \to \textit{snores}_3) \\
&= P(\mathrm{S} \to \mathrm{NP}\ \mathrm{VP}) P(\mathrm{NP} \to \textit{the man}) P(\mathrm{VP} \to \textit{snores})
\end{aligned}
$$

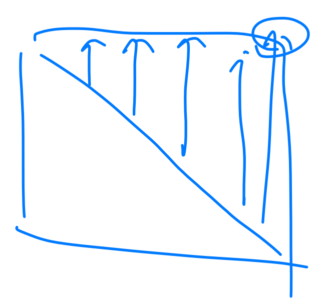
CYK using the scores:

on the diagonal

$$s_{\text{best}}(i, i+1) = \max_l s(i, i+1, l)$$

on other cells

$$
\begin{aligned}
s_{\text{best}}(i,j) &= \max_l s(i,j,l) \\
&+ \max_k [s_{\text{best}}(i,k) + s_{\text{best}}(k,j)]
\end{aligned}
$$



optimal Tree