CS224n : #3 : Dedendency Parsing written part

3/31/2021
at NYU-SH

1. [Machine Learning & Neural Network] (8 point)

(a)

(i) the momentum is such that

$$\begin{cases} m_{n+1} \longleftarrow \beta_1 \cdot m_n + (1-\beta_1) \nabla_\theta J_{minibatch}(\theta) \\ \theta_{n+1} \longleftarrow \theta_n - \alpha \cdot m_{n+1} \end{cases}$$

Intuitively, using "m" stops the updates from varying too much because each min-batch only contribute / change the momentum by a factor of $(1-\beta_1)$, with usually β_1 set to 0.9. So, if it so happens that a gradient of mini-batch deviates from the true whole-batch gradient too much, using momentum we would still roughly be on the right track.

(ii) the adaptive learning rates is such that

$$\begin{cases} m_{n+1} \leftarrow \beta_1 \cdot m_n + (1-\beta_1) \cdot \nabla_\theta J_{minibatch}(\theta) \\ V_{n+1} \leftarrow \beta_2 \cdot V_n + (1-\beta_2) \cdot (\nabla_\theta J_{minibatch}(\theta) \odot \nabla_\theta J_{minibatch}(\theta)) \\ \theta_{n+1} \leftarrow \theta_n - \alpha \odot m_{n+1} / \sqrt{V_{n+1}} \end{cases}$$

parameters that are ① previously very small in gradient
② variance is stable, stay more or less same in momentum

gets larger updates. This can help handle with sparse gradient.

(b)

$$h_{drop} = \gamma \cdot d \circ h$$

where $d \in \{0,1\}^{D_n}$ is mask vector, each entry with P_{drop} being 0, and $(1-P_{drop})$ being 1.

(i) to have $\mathbb{E}_{P_{drop}}[h_{drop}]_i = h_i$

$$\Rightarrow p_{drop} \cdot 0 + (1-p_{drop}) \cdot \gamma \cdot h_i = h_i$$

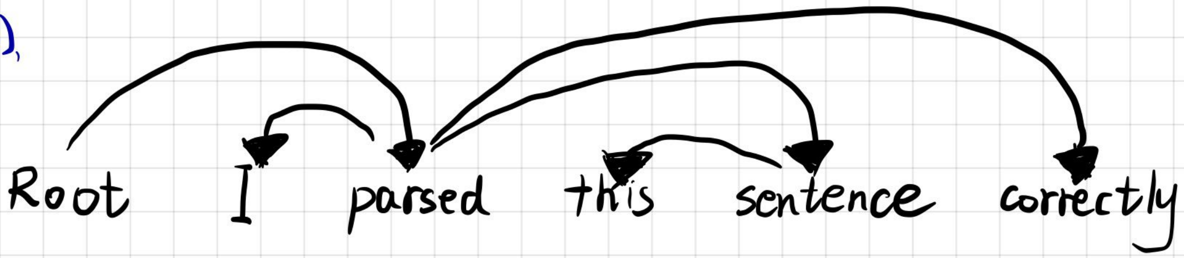
$$\Rightarrow \gamma = \frac{1}{1-p_{drop}}$$

(ii) apply dropout during training is a good way to do regularization, prevent overfitting

but in evaluation, we don't want a "random" result, so we don't apply dropout.

2. [Neural Translation-Based Dependency Parsing] (42 point)

(a)



Stack	Buffer	New dependency	Transition
[Root]	[I, parsed, this, sentence, corr]		Initial Configuration
[Root, I]	[parsed, this, sentence, corr]		Shift
[Root, I, parsed]	[this, sentence, corr]		Shift
[Root, parsed]	[this, sentence, corr]	parsed → I	Left-Arc
[Root, parsed, this]	[sentence, corr]		Shift
[Root, parsed, this, sent]	[correctly]		Shift
[Root, parsed, sent]	[correctly]	sentence → this	Left-Arc
[Root, parsed]	[correctly]	parsed → sent	Right-Arc
[Root, parsed, correctly]	[]		Shift
[Root, parsed]	[]	parsed → correctly	Right-Arc
[Root]	[]	Root → parsed	Right-Arc

(b). A sentence with n words with be parsed in 2n steps, because every word will be shifted into Stack once, and pop out by either left-Arc or Right-Arc once.

(c) code part

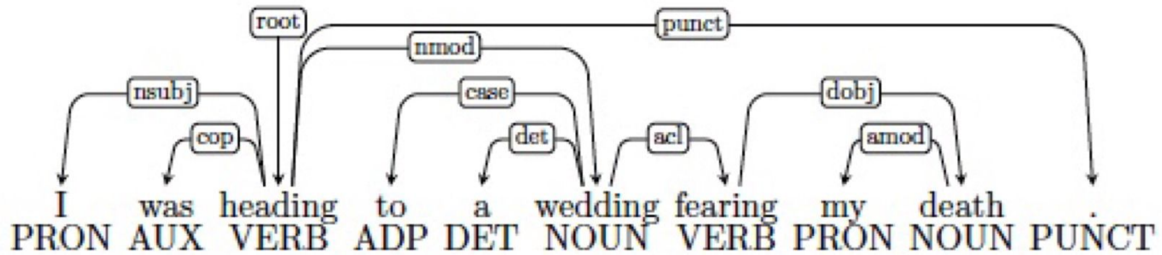
(d) code part

(e) code part

(f)

(i)

i.

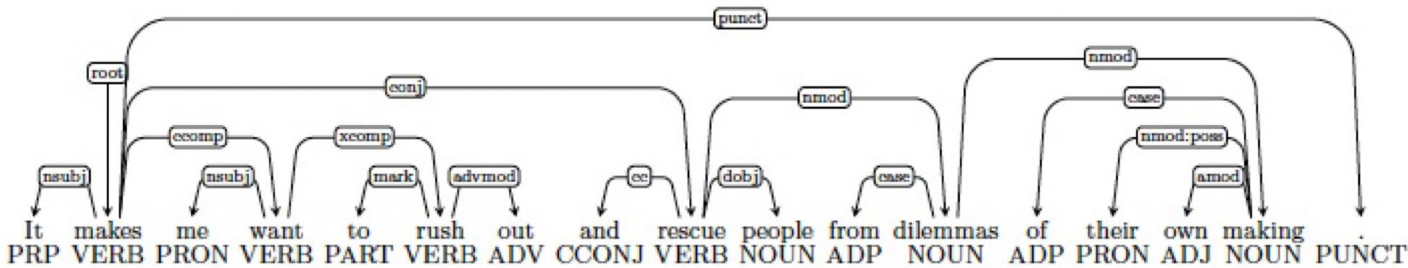


Error type: Verb Phrase Attachment Error

Incorrect dependency: wedding ⟶ fearing

Correct dependency: heading ⟶ fearing

(ii) ii.

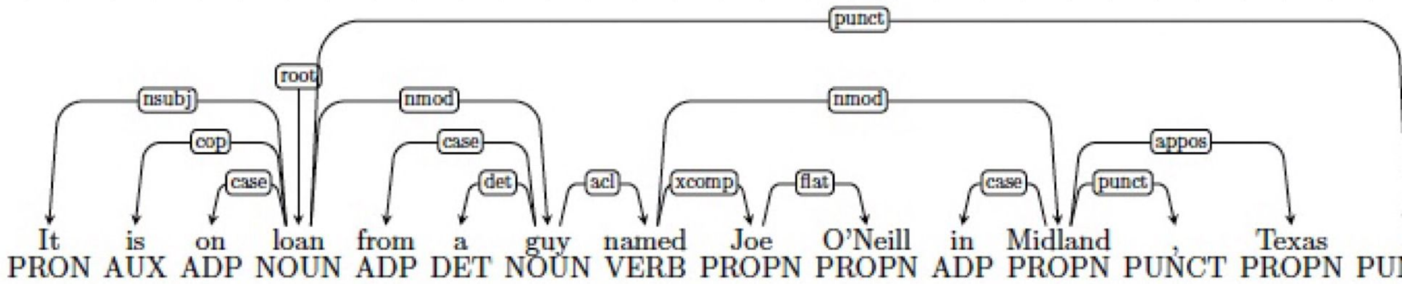


Error type: Coordination Attachment Error

Incorrect dependency: makes ⟶ rescue

correct dependency: rush ⟶ rescue

(iii)

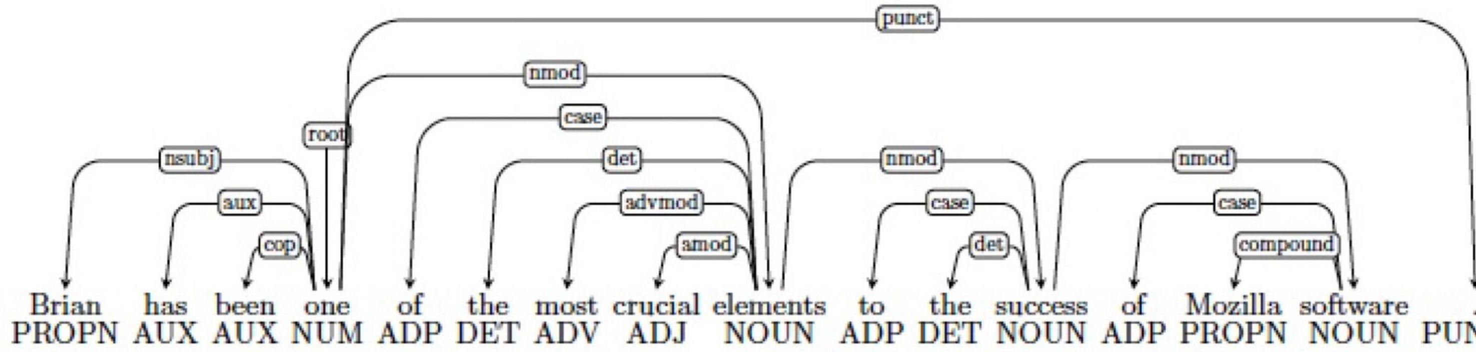


Error type : Prepositional Phrase Attachment Error

Incorrect dependency: named ⟶ Midland

correct dependency: guy ⟶ Midland

(iv)



Error type : Modifier Attachment Error

Incorrect dependency: elements ⟶ most

correct dependency: crucial ⟶ most

#

End of hw3