SUID:06349270

Answers for CS 224n Assignment #3: Dependency Parsing

1. Machine Learning & Neural Networks (8 points)
(a)

i. (2 points) First, Adam uses a trick called momentum by keeping track of m, a rolling average of the gradients:

$$m \leftarrow \beta_1 m + (1 - \beta_1) \nabla_\theta J_{minibatch}(\theta)$$
$$\theta \leftarrow \theta - \alpha m$$

where β_1 is a hyperparameter between 0 and 1 (often set to 0.9). Briefly explain (you don't need to prove mathematically, just give an intuition) how using m stops the updates from varying as much and why this low variance may be helpful to learning, overall.

Answer:

The momentum m is using exponentially weighted averages, which β_1 is weight of approximately last $\frac{1}{1-\beta_1}$ average minibatch gradients which is the m on the right side of the equation, $1-\beta_1$ is the weight of $\nabla_\theta J_{minibatch}(\theta)$, since we often set β_1 close to 1(often use0.9), so it gives most of the weight(0.9 in this case) to β_1 m, less weight (0.1 in this case)on $(1 - \beta_1)\nabla_\theta J_{minibatch}(\theta)$, so it makes each update will be mostly the same as the previous one, so θ update smoothly rather than varying as much.

With low variance of θ will reduce oscillation, make the update of θ more straight toward to the optima. Then the the model will converge faster, it also reduce the possibility for model being diverge when using larger learning rate α .

ii. (2 points) Adam also uses adaptive learning rates by keeping track of v, a rolling average of the magnitudes of the gradients:

$$m \leftarrow \beta_1 m + (1 - \beta_1) \nabla_\theta J_{minibatch}(\theta)$$
$$v \leftarrow \beta_2 v + (1 - \beta_2)(\nabla_\theta J_{minibatch}(\theta)$$
$$\theta \leftarrow \theta - \alpha m / \sqrt{v}$$
$$\nabla_\theta J_{minibatch}(\theta))$$

where and / denote elementwise multiplication and division (so z z is elementwise squaring) and β_2 is a hyperparameter between 0 and 1 (often set to 0.99). Since Adam divides the update by $\sqrt{v}$, which of the model parameters will get larger updates? Why might this help with learning?

Answer:

The smallest average gradients model parameters will get the larger updates.

The smallest average gradients model parameters are in the plateau area of the cost function, the moving is slow in the area of plateau during so the training process will be slow, if we could make those parameters in plateau area get larger updates will helping them move out of plateau quickly.

SUID:06349270

(b) (4 points) Dropout 3 is a regularization technique. During training, dropout randomly sets units
in the hidden layer h to zero with probability p drop (dropping different units each minibatch), and
then multiplies h by a constant γ. We can write this as
h drop = γd ◦ h
where d ∈ {0, 1} D h (D h is the size of h) is a mask vector where each entry is 0 with probability
p drop and 1 with probability (1 − p drop). γ is chosen such that the expected value of h drop is h:
E p drop [h drop] i = h i
for all i ∈ {1, . . . , D h }.

i. (2 points) What must γ equal in terms of p drop ? Briefly justify your answer.

Answer:

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

Justify:

$$E_{Pdrop}[h_{drop}]_i = E_{Pdrop}[\gamma d \circ h]_i = (1 - p_{drop}) h_i \gamma$$

since we want $E_{Pdrop}[h_{drop}]_i = h_i$ **means we want** $(1 - p_{drop}) h_i \gamma = h_i$ **, so** $\gamma = \frac{1}{1 - p_{drop}}$

Another way of understanding :

After perform d ◦ h(suppose h' = d ◦ h,), $p_{drop} * 100\%$ **of elements in h zeroed out. So in order to not reduce the expected value of** h_{drop} **, we have to "invert" the dropout by perform** $\frac{h'}{1 - p_{drop}}$ **this will bump backup about by** $p_{drop} * 100\%$ **for h that has lost.**

ii. (2 points) Why should we apply dropout during training but not during evaluation?

Answer:

If we apply dropout during evaluation, it will make the output to be random, which we don't want . It adds noise to the prediction. The evaluation will not be accrue.

SUID:06349270

2. Neural Transition-Based Dependency Parsing (42 points)

(a) (6 points) Go through the sequence of transitions needed for parsing the sentence "I parsed this sentence correctly". The dependency tree for the sentence is shown below. At each step, give the configuration of the stack and buffer, as well as what transition was applied this step and what new dependency was added (if any). The first three steps are provided below as an example.

Stack	Buffer	New dependency	Transition
[ROOT]	[I, parsed, this, sentence, correctly]		Initial Configuration
[ROOT, I]	[parsed, this, sentence, correctly]		SHIFT
[ROOT, I, parsed]	[this, sentence, correctly]		SHIFT
[ROOT, parsed]	[this, sentence, correctly]	parsed→I	LEFT-ARC

Answer:

Stack	Buffer	New dependency	Transition
[ROOT]	**[I, parsed, this, sentence, correctly]**		**Initial Configuration**
[ROOT, I]	**[parsed, this, sentence, correctly]**		**SHIFT**
[ROOT,I, parsed]	**[this , sentence ,correctly]**		**SHIFT**
[ROOT, parsed]	**[this, sentence, correctly]**	**parsed → I**	**LEFT-ARC**
[ROOT, parsed, this]	**[sentence, correctly]**		**SHIFT**
[ROOT, parsed, this,sentence]	**[correctly]**		**SHIFT**
[ROOT, parsed, sentence]	**[correctly]**	**sentence → this**	**LEFT-ARC**
[ROOT, parsed]	**[correctly]**	**parsed → sentence**	**RIGHT-ARC**
[ROOT,parsed,correctly]	**[]**		**SHIFT**
[ROOT, parsed]	**[]**	**parsed → correctly**	**RIGHT-ARC**
[ROOT]	**[]**	**ROOT→parsed**	**RIGHT-ARC**

SUID:06349270

(b) (2 points) A sentence containing n words will be parsed in how many steps (in terms of n)? Briefly explain why.

Answer:

2n if we do not consider Initial Configuration as a step.

Reason: for each word in sentence will have to push from buffer to stack, the cost of the step is n, then each word pushed to stack have to perform either LEFT-ARC or RIGHT-ARC, and all words will be in stack once, so the cost of total xx-ARC(xx = LEFT or RIGHT) step is n. so it`s 2n in totally steps for all SHIFT, ARC operations.

SUID:06349270

2. Neural Transition-Based Dependency Parsing (42 points)

(e) Report the best UAS your model achieves on the dev set and the UAS it achieves on the test set.

Answer:

Best UAS of dev set: 73.38
Best UAS of test set: 89.50

Dev:

```
Epoch 10 out of 10
100%|██████████| 48/48 [00:07<00:00,  6.32it/s]
Average Train Loss: 0.14279178700720271
Evaluating on dev set
  0%|          | 0/500 [00:00<?, ?it/s]- dev UAS: 73.38
New best dev UAS! Saving model.
125250it [00:00, 8325196.92it/s]
```

Test:

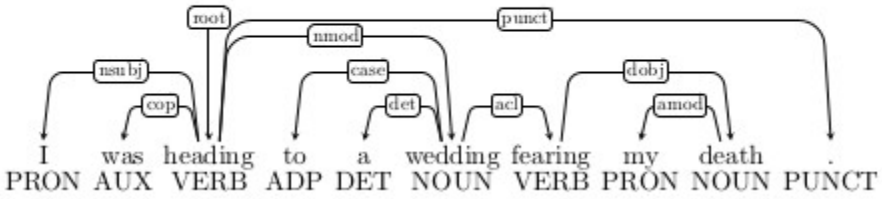
```
================================================================================
TESTING
================================================================================
Restoring the best model weights found on the dev set
Final evaluation on test set
2919736it [00:00, 62764320.45it/s]
- test UAS: 89.50
Done!

Process finished with exit code 0
```

SUID:06349270

(f)(i)

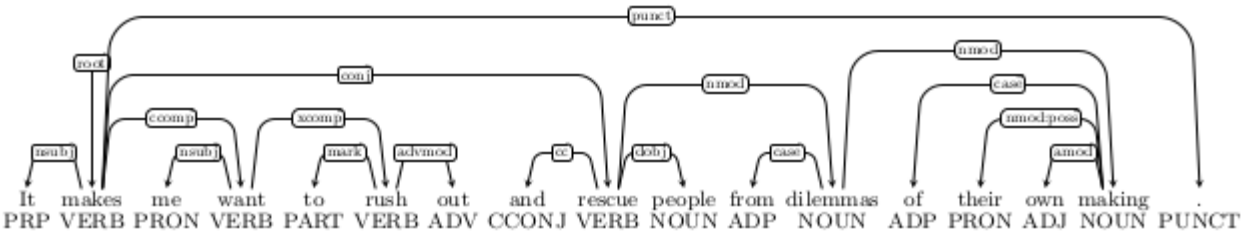
i.



Answer:
Error Type: Verb Phrase Attachment Error
Incorrect dependency: wedding → fearing
Correct dependency : heading → fearing

(ii)

ii.

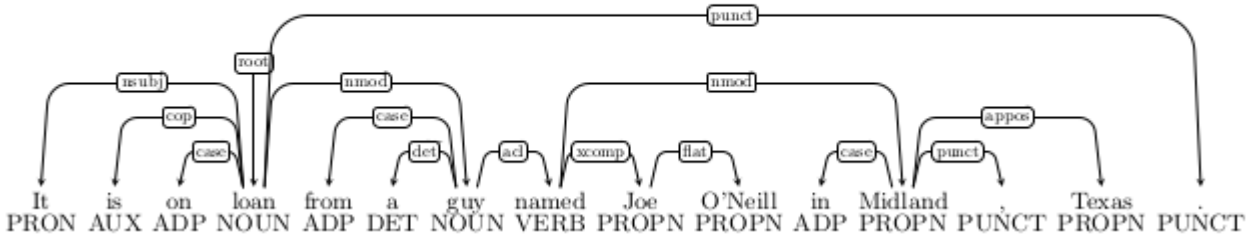


Answer:
Error Type: Coordination Attachment Error
Incorrect dependency: makes → rescue
Correct dependency: rush → rescue

SUID:06349270

(iii)

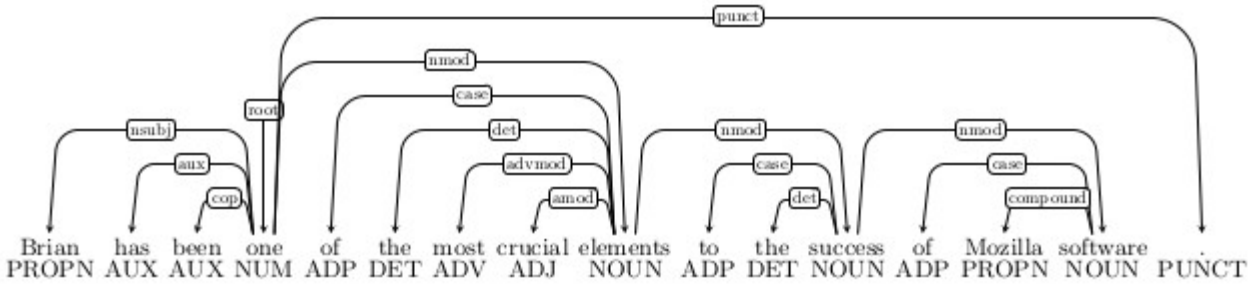
iii.



Answer:
Error Type: Prepositional Phrase Attachment Error
Incorrect Dependency: named → Midland
Correct Dependency: guy → Midland

(iv)

iv.



Answer:
Error Type: Modifier Attachment Error
Incorrect dependency: elements → most
correct dependency: crucial → most