Assignment 3 Notes

1 Machine Learning & Neural Network

(a)

(i) (2 points)

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

Briefly explain (you don't need to prove mathematically, just give an intuition) how using momentum stops the updates from varying as much and why this low variance may be helpful to learning, overall.

- Momentum maintains the change in the original position and uses it in the subsequent calculation of change in the direction, which adds history to the parameter update based on the gradient encountered in the previous updates. This can reduce effects of the short-time change in direction of gradient on the direction of updating parameters because it reduces the effect of newly calculated gradient.
- Such low variance can play an important role where the objective function has a large amount of curvature, which indicates the gradient may change a lot over relatively small regions of the search space. Momentum increases the difficulty to change the direction of search by changed gradients in small regions as the changes are accumulated through the time.

(ii)

Adam also uses adaptive learning rates by keeping track of v, a rolling average of the magnitudes of the gradients:

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

where ⊙â€‹ and / denote elementwise multiplication and division (so $z \odot z$ 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?*

- the parameters which have lower magnitudes of gradients will get larger updates.
- This change assigns greater learning rates to the parameters which are less updated and more information can be extracted from them.

(b)

Dropout is a regularization technique. During training, dropout random sets units in the hidden layer $\vec{h}$ (their outputs) to zero with probability p_{drop}â€‹ and then multiplies $\vec{h}$ by a constant γâ€‹.

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

where $d \in \{0,1\}^{D_h}$ 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 $\vec{h}_{drop}$ is $\vec{h}$:

$$\mathbb{E}_{p_{drop}}[\vec{h}_{drop}]_i = h_i$$

for all $i \in \{1, \dots, D_h\}$

(i)

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

- $$\begin{aligned} \mathbb{E}_{p_{drop}}[\vec{h}_{drop}]_i &= h_i \gamma \mathbb{E}(d_i) \\ &= h_i \gamma (0 \cdot p_{drop} + 1 \cdot (1 - p_{drop})) \\ &= h_i \gamma (1 - p_{drop}) = h_i \end{aligned}$$
- we can arrive at $\gamma(1 - p_{drop}) = 1$ so $\gamma = \frac{1}{1-p_{drop}}$

(ii)

Because the some of the outputs of the dropout layers can be randomly dropped, the training process is made nosier. In turn the algorithm is forced to be more robust and have better generalization ability to adapt to such changes.

However, during evaluation, there shouldn't be random effect of dropping outputs because this may resulted in inconsistent outputs. (The evaluation is based on the quality of outputs)

2 Neural Transition-Based Dependency Parsing

(a)

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 $\rightarrow$ I	LEFT-ARC
[ROOT, parsed, this]	[sentence, correctly]		SHIFT
[ROOT, parsed, this, sentence]	[correctly]		SHIFT
[ROOT, parsed, sentence]	[correctly]	sentence $\rightarrow$ this	LEFT-ARC
[ROOT, parsed]	[correctly]	parsed $\rightarrow$ sentence	RIGHT-ARC
[ROOT,parsed,correctly]	$\emptyset$		SHIFT
[ROOT,parsed]	$\emptyset$	parsed $\rightarrow$ correctly	RIGHT-ARC
[ROOT]	$\emptyset$	ROOT $\rightarrow$ parsed	RIGHT-ARC

(b)

$O(2n)$

Each word will be shifted to the stack once and then removed from the stack once. Each of these 2 operations takes one step. So the total steps should be $2n$

(c)

Implement the `__init__` and `parse_step` functions in the `PartialParse` class in `parser_transitions.py` This implements the transition mechanics your parser will use.

You can run basic (non-exhaustive) tests by running `python parser_transitions.py part_c`.

(d)

Our network will predict which transition should be applied next to a partial parse. We could use it to parse a single sentence by applying predicted transitions until the parse is complete. However, neural networks run much more efficiently when making predictions about batches of data at a time (i.e., predicting the next transition for any different partial parses simultaneously). We can parse sentences in minibatches with the following algorithm.

(e)

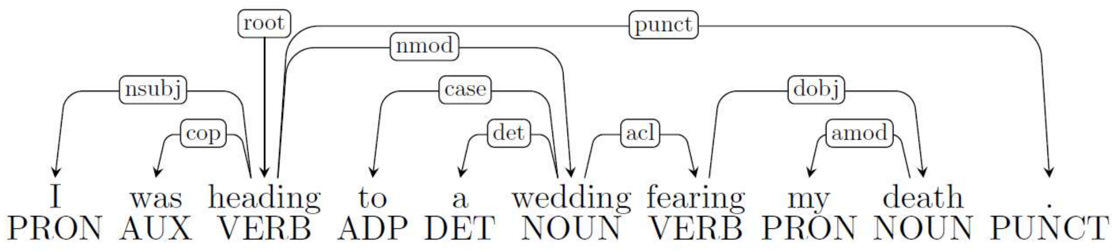
- The best UAS on dev set is 88.48
- its UAS on test set is 88.71

```
================================================================================
TESTING
================================================================================
Restoring the best model weights found on the dev set
Final evaluation on test set
2919736it [00:00, 39064776.05it/s]
- test UAS: 88.71
Done!
```

(f)

the definition of errors are noted in the lecture notes

i

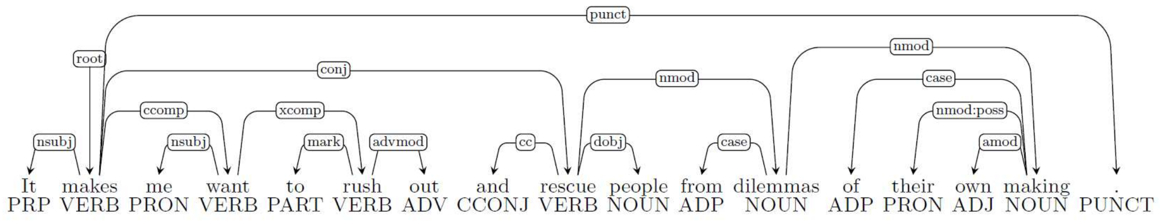


I was heading to a wedding fearing my death.

- Error type:
  - Modifier Attachment Error
- Incorrect Dependency:
  - $wedding \rightarrow fearing$
- Correct Dependency:
  - $heading \rightarrow fearing$

ii

It makes me want to rush out and rescue people from dilemmas of their own making.

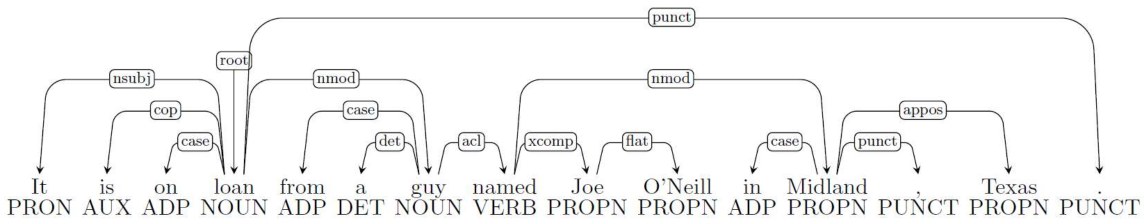


- Error type:
  - Coordination Attachment Error
- Incorrect Dependency:
  - $makes \rightarrow rescue$
- Correct Dependency:

    - $rush \rightarrow rescue$

iii

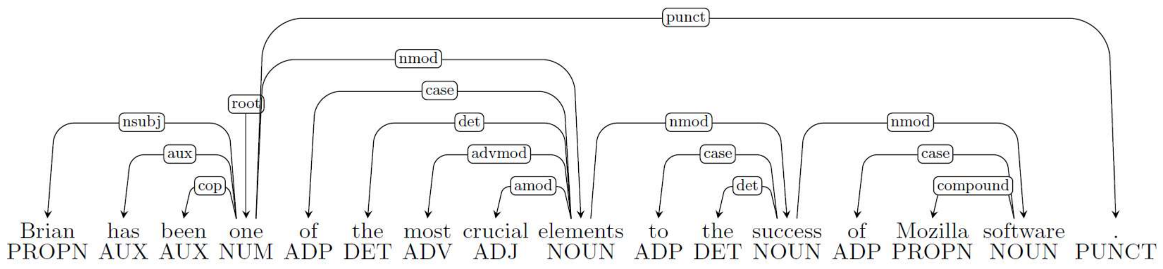
It is an loan from a guy named Joe O'Neill in Midland, Texas



- Error type:
    - Prepositional Phrase Attachment Error
- Incorrect Dependency:
    - $named \rightarrow Midland$
- Correct Dependency:
    - $guy \rightarrow Midland$

iv

Brian has been one of the most crucial elements to the success of Mozilla software.



- Error type:
    - Modifier Attachment Error
- Incorrect Dependency:
    - $elements \rightarrow most$
- Correct Dependency:
    - $crucial \rightarrow most$