Assignment 3: Dependency Parsing

1. Machine Learning & Neural Networks

(a) Adam Optimizer

i. Briefly explain how momentum (by keeping track of parameter m) stops the updates from varying as much and why this low variance may be helpful to learning.

With simple SGD, the optimization process can be too oscillated and unefficient, i.e. informally speaking, update steps goes zig-zag. However, if we can smooth each update, we can reduce the variance and make the optimation process more efficient. There are several ways to perform smooth function, the benefit of exponential average/ weighted moving average lies in 1) more efficient on memory usage 2) easy to code and implement. In this way, we can control the update by changing β_1. Note that when $\beta_1 = 0$, this is essentially SGD.

ii. Adam extends the idea of momentum with the track of adaptive learning rates by keeping track of $\boldsymbol{v}$, a rolling average of the magnitudes of the gradients. Since Adam divides the update by $\sqrt{\boldsymbol{v}}$, which of the model parameters will get larger updates? Why might this help with learning?

Since $\boldsymbol{v}$ is a rolling average of the magnitude of the gradient, parameters of the model with smaller magnitude, i.e. with small $\boldsymbol{v}$ entry, will get larger updates. Essentially, we are cutting out the smaller amount from such parameters during the updates because we devide them by their magnitude. At the same time, the parameters with bigger magnitude will have smaller updates. This approach have two benefits that 1) get recent stagnant parameters update more efficiently along their axis and in turn expedite the optimization 2) smooth some "large" updates to get robust performance.

(b) Dropout

i. What must γ equal in tems of p_{drop}? Briefly justify your answer or show your math derivation using the equation given above.

Given that $\boldsymbol{h}_{drop} = \gamma \boldsymbol{d} \cdot \boldsymbol{h}$, and $\mathbb{E}_{p_{drop}}[\boldsymbol{h}_{drop}]_i = h_i$ for all $i \in 1, \ldots, D_h$, it is straightforward to show that $\gamma = \frac{1}{1-p_{drop}}$. Intuitively, in training part we dropout some units, but in the test part there is no dropout (as this only increase noises). As a result, we have to fix the units that to make the outputs of the training and testing are similar, and the fix factor is γ.

ii. Why should dropout be applied during training? Why should dropout NOT be applied during evaluation?

Why use dropout: dropout is a useful regulization technique to deal with overfitting problem. Intuitively, with dropout, we make the neural net more robust to different dataset (especially the datasets that it never "see" before).

Why NOT use it during evaluation: when we evaluate the neural net, we are concerned with how well the model is with unseen data. In this case, if we use dropout, we are "thinning" the net which in many cases only increase the noise to the predictions and dampen the accuracy of the model, and we cannot fairly assess the generalization power of the model.

2. Neural Transition-based Dependency Parsing

(a) Go through the sequence of transitions needed for parsing the sentence "I parsed this sentence correctly".

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	LIFT-ARC
[ROOT, parsed]	[correctly]	parsed→sentence	RIGHT-ARC
[ROOT, parsed, correctly]	[]		SHIFT
[ROOT, parsed]	[]	parsed→correctly	RIGHT-ARC
[ROOT]	[]	ROOT→parsed	RIGHT-ARC

(b) A sentence containing n words will be parsed in how many steps (in terms of n)?

O(n). In particular, each word must goes from buffer to stack exactly once, so we have n SHIFT steps. Similarly, each word must be removed from the stack exactly once, thus we have n LEFT-ARC or RIGHT-ARC steps. In total, this gives 2n time, i.e. O(n).

(f)

i.

- Error type: Verb Phrase Attachment Error
- Incorrect dependency: wedding → fearing
- Correct dependency: heading → fearing

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