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

where β_1 is a hyperparameter between 0 and 1 (often set to 0.9). Briefly explain in 2–4 sentences (you don't need to prove mathematically, just give an intuition) how using $\mathbf{m}$ stops the updates from varying as much and why this low variance may be helpful to learning, overall.

By taking the average gradient we make sure that there is no one gradient makes us change our direction sharply. we can look at m as momentum to the the gradient that will help us pass local minimum.

ii. (2 points) Adam extends the idea of *momentum* with the trick of *adaptive learning rates* by keeping track of $\mathbf{v}$, a rolling average of the magnitudes of the gradients:

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

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

Dividing by the root of v makes the updates after large steps very small and the updates after small steps bigger.this helps the function to escape stacking in local minima or missing minima between sharp gradients.

(b) (4 points) Dropout[3] is a regularization technique. During training, dropout randomly sets units in the hidden layer $\mathbf{h}$ to zero with probability p_{drop} (dropping different units each minibatch), and then multiplies $\mathbf{h}$ by a constant γ. We can write this as:

$$\mathbf{h}_{\text{drop}} = \gamma \mathbf{d} \odot \mathbf{h}$$

where $\mathbf{d} \in \{0,1\}^{D_h}$ (D_h is the size of $\mathbf{h}$) is a mask vector where each entry is 0 with probability p_{drop} and 1 with probability $(1 - p_{\text{drop}})$. γ is chosen such that the expected value of $\mathbf{h}_{\text{drop}}$ is $\mathbf{h}$:

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

(2)

for all $i \in \{1, \ldots, D_h\}$.

i. (2 points) What must γ equal in terms of p_{drop}? Briefly justify your answer or show your math derivation using the equations given above.

ii. (2 points) Why should dropout be applied during training? Why should dropout **NOT** be applied during evaluation? (Hint: it may help to look at the paper linked above in the write-up.)

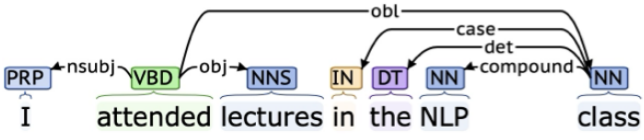
i. $\mathbb{E}_{p_{drop}}[h_{drop}]_i = p_{drop} \cdot 0 + (1-p_{drop}) \cdot h_{drop} = (1-p_{drop})\gamma d \odot h_i =$ (1)

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

(2) $h_i(1-p_{drop})$

ii. Dropout increase robustness so it won't be held only by few neurons. When training we are trying to get most of the information from the neurons so we drop some of them to make the others activate. When evaluating we want to use every information we got so we don't drop anything.

(a) (4 points) Go through the sequence of transitions needed for parsing the sentence *"I attended lectures in the NLP class"*. 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, attended, lectures, in, the, NLP, class]		Initial Configuration
[ROOT, I]	[attended, lectures, in, the, NLP, class]		SHIFT
[ROOT, I, attended]	[lectures, in, the, NLP, class]		SHIFT
[ROOT, attended]	[lectures, in, the, NLP, class]	attended→I	LEFT-ARC

Stack	Buffer	New dependency	Transition
[ROOT]	[I, attended, lectures, in, the, NLP, class]	-	Initial configuration
[ROOT, I]	[attended, lectures, in, the, NLP, class]	-	Shift
[ROOT, I, attended]	[lectures, in, the, NLP, class]	-	Shift
[ROOT, attended]	[lectures, in, the, NLP, class]	attended → I	L-Arc
[ROOT, attended, lectures]	[in, the, NLP, class]	-	Shift
[ROOT, attended]	[in, the, NLP, class]	attended → lectures	R-Arc
[ROOT, attended, in]	[the, NLP, class]	-	Shift
[ROOT, attended, in, the, NLP, class]	[]	-	Shift
[ROOT, attended, class]	[]	class → NLP, the, in	L-Arc
[ROOT, attended]	[]	attended → class	R-Arc
[ROOT]	[]	ROOT → attended	R-Arc

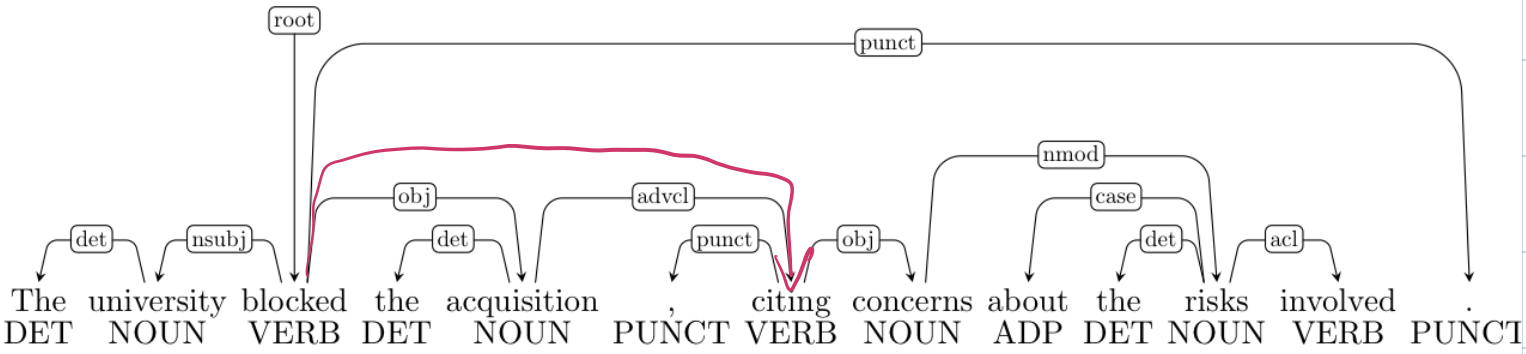
(b) (2 points) A sentence containing n words will be parsed in how many steps (in terms of n)? Briefly explain in 1–2 sentences why.

(b) 2n because there are n steps for shifting the sentence and one step for parsing each word (L-arc or R-arc)

- **Report the best UAS your model achieves on the dev set and the UAS it achieves on the test set in your writeup.**

Using 500 as hidden size I got 89.24 on dev UAS, and 90.03 on test UAS

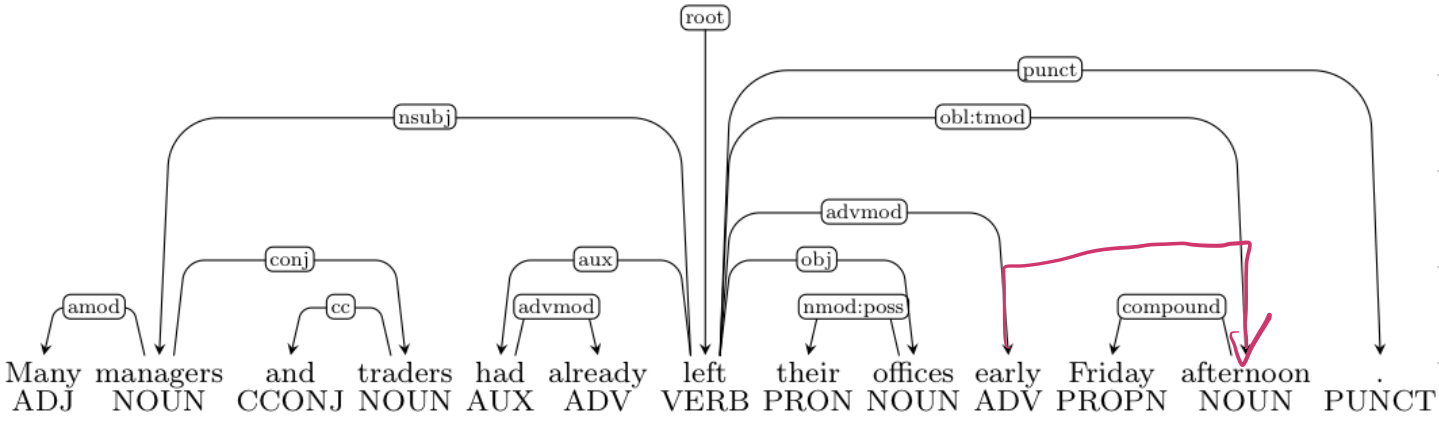
i.



i. Error type: verb phrase error

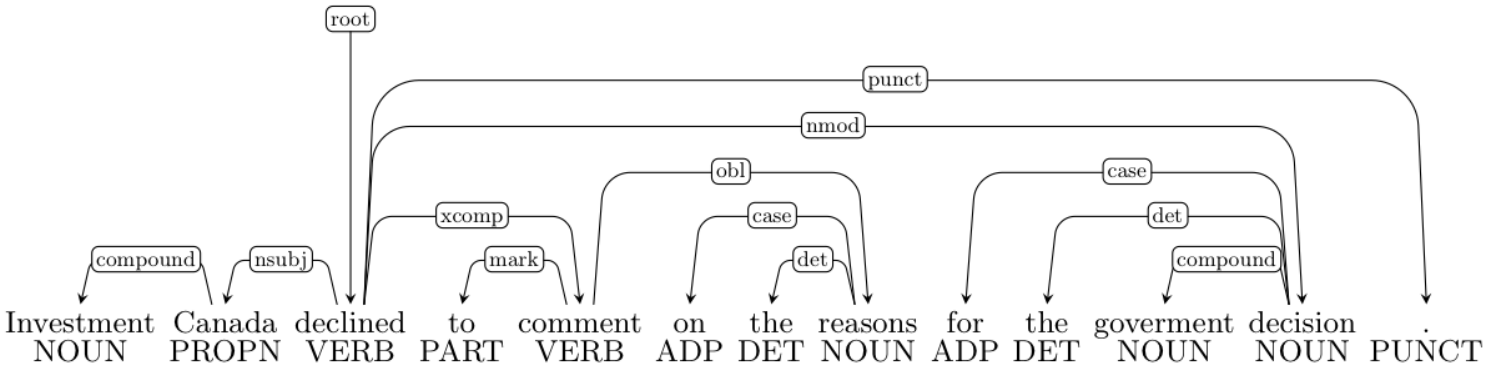
- Incorrect dependency: acquisition → citing
- Correct dependency: blocked → citing

ii.



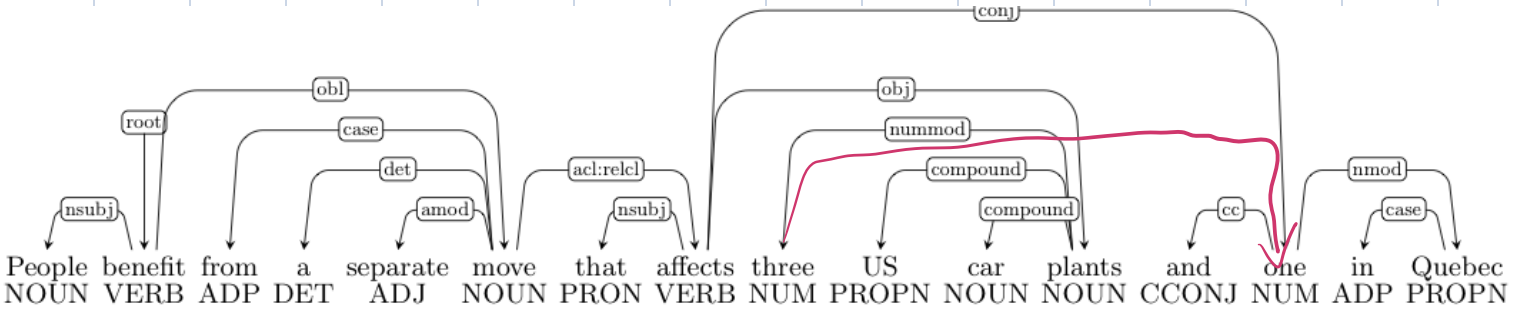
ii. Error type: modifier attachment error

- Incorrect dependency: left → early
- Correct dependency: afternoon → early



iii. Error type: prepositional attachment error

- Incorrect dependency: declined → decision
- Correct dependency: reasons → decision



iv. Error type: coordination attachment error

- Incorrect dependency: affects → one
- Correct dependency: plants → one

(g) (2 points) Recall in part (e), the parser uses features which includes words and their part-of-speech (POS) tags. Explain the benefit of using part-of-speech tags as features in the parser?

POS are useful information that provide features for the parser to avoid disambiguate between possible parses like we have seen (f), by the grammatical function of words in the sentence.