Assignment 3 Written Solutions

(a) (4 points) Adam Optimizer

Recall the standard Stochastic Gradient Descent update rule:

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

where $t+1$ is the current timestep, $\boldsymbol{\theta}$ is a vector containing all of the model parameters, ($\boldsymbol{\theta}_t$ is the model parameter at time step t, and $\boldsymbol{\theta}_{t+1}$ is the model parameter at time step $t+1$), J is the loss function, $\nabla_{\boldsymbol{\theta}} J_{\text{minibatch}}(\boldsymbol{\theta})$ is the gradient of the loss function with respect to the parameters on a minibatch of data, and α is the learning rate. Adam Optimization[1] uses a more sophisticated update rule with two additional steps.[2]

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

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

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.

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?

(i) The gradient could be a very large or very small number. If the gradient is a very large number, the update may miss the optimal value. If the gradient is a very small number, the convergence to the optimal value will be small. Instead of using the gradient directly, the algorithm updates the weights based on the average (or the moving average) of the previous gradients. This moves the update into a stable and more accurate direction towards the optimal value.

(ii) The parameters whose gradients have small magnitudes. This allows adaptive learning rates $\frac{\alpha}{\sqrt{v}}$ for each individual parameter rather than a constant α for all parameters. This guides the parameters into a faster and more efficient convergence.

(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 $\mathbf{h}$:

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

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) The expected value of h_{drop} is:

$$\mathbb{E}_{p_{drop}}[\boldsymbol{h}_{\text{drop}}] = \mathbb{E}_{p_{drop}}[\gamma \boldsymbol{d} \odot \boldsymbol{h}]$$
$$= \gamma \mathbb{E}_{p_{drop}}[\boldsymbol{d} \odot \boldsymbol{h}] \text{ ,since γ is a constant}$$
$$= \gamma \mathbb{E}_{p_{drop}}[\boldsymbol{d}] \odot \boldsymbol{h} \text{ ,since h is not affected by dropout}$$
$$= \gamma(p_{drop} \times 0 + (1 - p_{drop}) \times 1) \odot \boldsymbol{h}$$
$$= \gamma(1 - p_{drop}) \odot \boldsymbol{h}$$

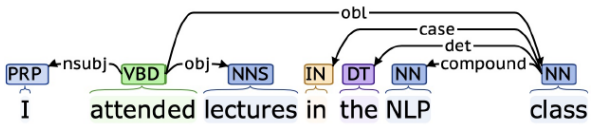
But we want $\mathbb{E}_{p_{drop}}[\boldsymbol{h}_{\text{drop}}] = \boldsymbol{h}$. So,

$$\boldsymbol{h} = \gamma(1 - p_{drop}) \odot \boldsymbol{h}$$
$$1 = \gamma(1 - p_{drop})$$
$$\gamma = \frac{1}{1 - p_{drop}}$$

(ii) It is applied during training to increase generalization through cancelling random features. This forces the algorithm to learn from different features at each cycle and not stick to certain features.

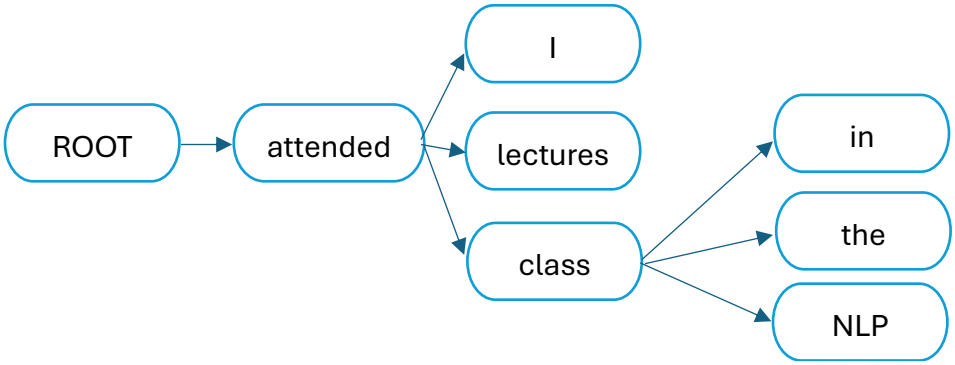
It is not applied during evaluation because we want the algorithm to make a prediction based on all features not a subset of them. Applying dropout during evaluation will cause undesirable and inconsistent results.

(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	LEFT-ARC
[ROOT, attended, lectures]	[in, the, NLP, class]		SHIFT
[ROOT, attended]	[in, the, NLP, class]	attended → lectures	RIGHT-ARC
[ROOT, attended, in]	[the, NLP, class]		SHIFT
[ROOT, attended, in, the]	[NLP, class]		SHIFT
[ROOT, attended, in, the, NLP]	[class]		SHIFT
[ROOT, attended, in, the, NLP, class]	[]		SHIFT
[ROOT, attended, in, the, class]	[]	class → NLP	LEFT-ARC
[ROOT, attended, in, class]	[]	class → the	LEFT-ARC
[ROOT, attended, class]	[]	class → in	LEFT-ARC
[ROOT, attended]	[]	attended → class	RIGHT-ARC
[ROOT]	[]	ROOT → attended	RIGHT-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.

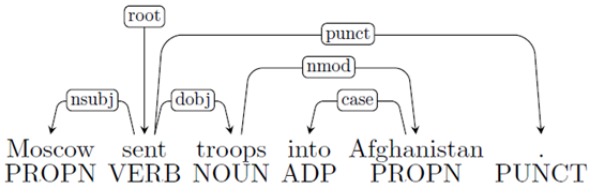
A sentence containing n words will be parsed in $2n$ steps. We need n steps to `SHIFT` the whole sentence and another n steps to determine the type of dependency (`LEFT-ARC` or `RIGHT-ARC`).

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

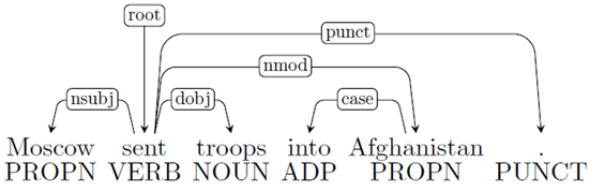
Best UAS on the dev set: **88.43%**

Best UAS on the test set: **89.23%**

(f) (12 points) We'd like to look at example dependency parses and understand where parsers like ours might be wrong. For example, in this sentence:



the dependency of the phrase *into Afghanistan* is wrong, because the phrase should modify *sent* (as in *sent into Afghanistan*) not *troops* (because *troops into Afghanistan* doesn't make sense, unless there are somehow weirdly some troops that stan Afghanistan). Here is the correct parse:



More generally, here are four types of parsing error:

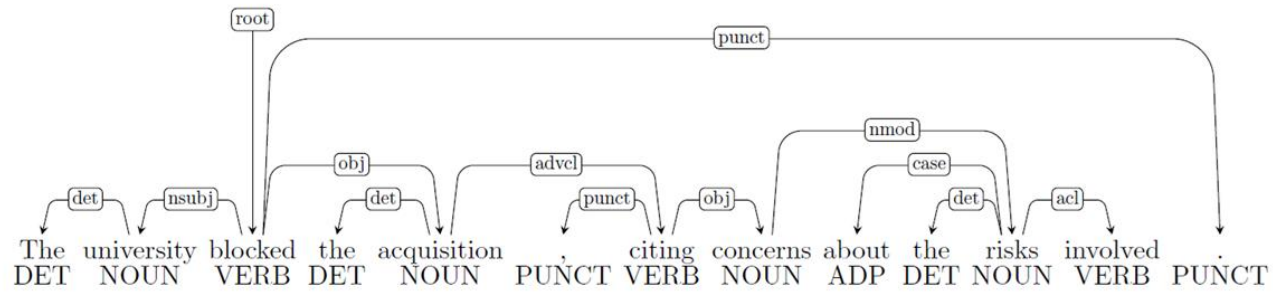
- **Prepositional Phrase Attachment Error**: In the example above, the phrase *into Afghanistan* is a prepositional phrase[5]. A Prepositional Phrase Attachment Error is when a prepositional phrase is attached to the wrong head word (in this example, *troops* is the wrong head word and *sent* is the correct head word). More examples of prepositional phrases include *with a rock*, *before midnight* and *under the carpet*.
- **Verb Phrase Attachment Error**: In the sentence *Leaving the store unattended, I went outside to watch the parade*, the phrase *leaving the store unattended* is a verb phrase[6]. A Verb Phrase Attachment Error is when a verb phrase is attached to the wrong head word (in this example, the correct head word is *went*).
- **Modifier Attachment Error**: In the sentence *I am extremely short*, the adverb *extremely* is a modifier of the adjective *short*. A Modifier Attachment Error is when a modifier is attached to the wrong head word (in this example, the correct head word is *short*).
- **Coordination Attachment Error**: In the sentence *Would you like brown rice or garlic naan?*, the phrases *brown rice* and *garlic naan* are both conjuncts and the word *or* is the coordinating conjunction. The second conjunct (here *garlic naan*) should be attached to the first conjunct (here *brown rice*). A Coordination Attachment Error is when the second conjunct is attached to the wrong head word (in this example, the correct head word is *rice*). Other coordinating conjunctions include *and*, *but* and *so*.

In this question are four sentences with dependency parses obtained from a parser. Each sentence has one error type, and there is one example of each of the four types above. For each sentence, state the type of error, the incorrect dependency, and the correct dependency. While each sentence should have a unique error type, there may be multiple possible correct dependencies for some of the sentences. To demonstrate: for the example above, you would write:

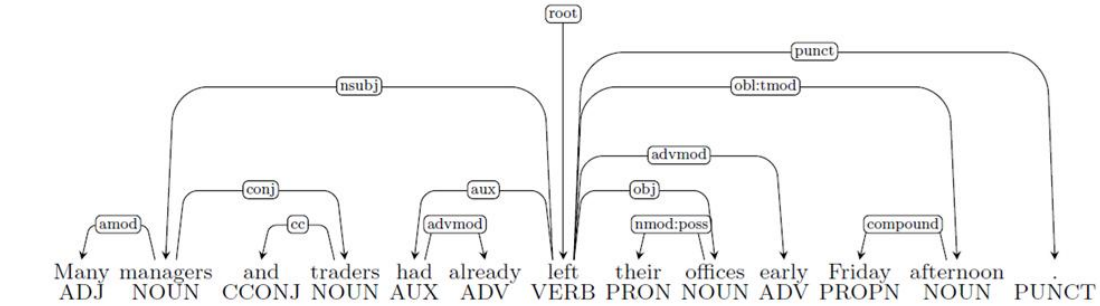
- **Error type**: Prepositional Phrase Attachment Error
- **Incorrect dependency**: troops → Afghanistan
- **Correct dependency**: sent → Afghanistan

Note: *There are lots of details and conventions for dependency annotation. If you want to learn more about them, you can look at the UD website:* `http://universaldependencies.org`[7] *or the short introductory slides at:* `http://people.cs.georgetown.edu/nschneid/p/UD-for-English.pdf`. *Note that you* ***do not*** *need to know all these details in order to do this question. In each of these cases, we are asking about the attachment of phrases and it should be sufficient to see if they are modifying the correct head. In particular, you* ***do not*** *need to look at the labels on the the dependency edges – it suffices to just look at the edges themselves.*

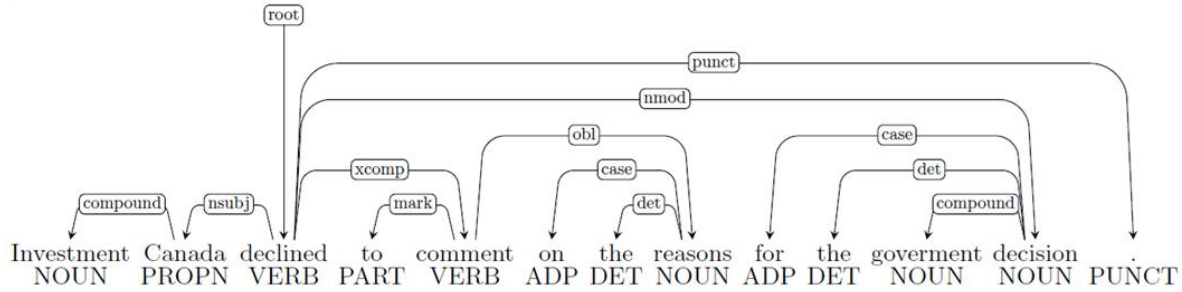
i.


root
punct
nmod
obj
advcl
case
det
nsubj
det
punct
obj
det
acl
The university blocked the acquisition , citing concerns about the risks involved .
DET NOUN VERB DET NOUN PUNCT VERB NOUN ADP DET NOUN VERB PUNCT

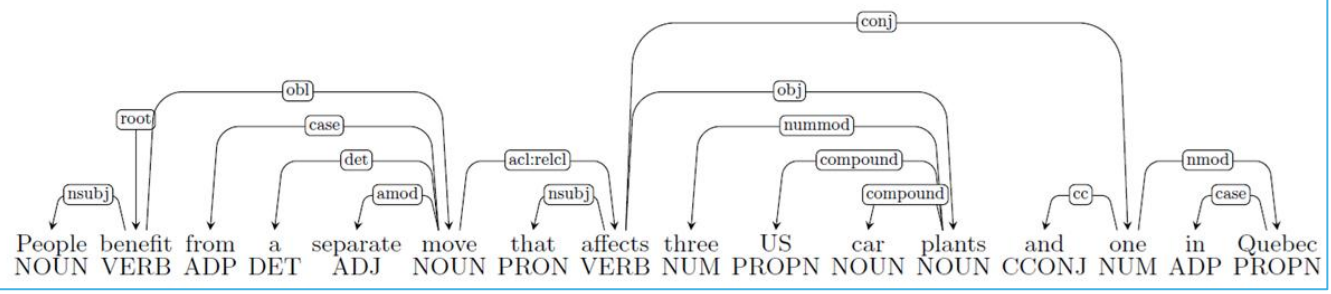
ii.


root
punct
nsubj
obl:tmod
advmod
conj
aux
obj
amod
cc
advmod
nmod:poss
compound
Many managers and traders had already left their offices early Friday afternoon .
ADJ NOUN CCONJ NOUN AUX ADV VERB PRON NOUN ADV PROPN NOUN PUNCT

iii.


root
punct
nmod
obl
case
xcomp
case
det
compound
nsubj
mark
det
compound
Investment Canada declined to comment on the reasons for the goverment decision .
NOUN PROPN VERB PART VERB ADP DET NOUN ADP DET NOUN NOUN PUNCT

iv.


conj
obl
obj
root
case
nummod
det
acl:relcl
compound
nmod
nsubj
amod
nsubj
compound
cc
case
People benefit from a separate move that affects three US car plants and one in Quebec
NOUN VERB ADP DET ADJ NOUN PRON VERB NUM PROPN NOUN NOUN CCONJ NUM ADP PROPN

(i) **Error Type:** Verb Phrase Attachment Error
Incorrect Dependency: acquisition → citing
Correct Dependency: blocked → citing

(ii) **Error Type:** Modifier Attachment Error
Incorrect Dependency: left → early
Correct Dependency: afternoon → early

(iii) **Error Type:** Prepositional Phrase Attachment Error
Incorrect Dependency: declined → decisions
Correct Dependency: reasons → decisions

(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?

- **Text Simplification:** Breaking complex sentences down into their constituent parts makes the material easier to understand and easier to simplify.
- **Information Retrieval:** Information retrieval systems are enhanced by point-of-sale (POS) tagging, which allows for more precise indexing and search based on grammatical categories.
- **Named Entity Recognition:** POS tagging helps to identify entities such as names, locations, and organizations inside text and is a precondition for named entity identification.
- **Syntactic Parsing:** It facilitates syntactic parsing, which helps with phrase structure analysis and word link identification.
- **Disambiguation:** Words like “play” can be a noun or verb. POS tagging helps identify the correct meaning based on context, using tag sets that define the possible tags for each word type and their contexts.