Hoorieh Sabzevari – 98412004

NLP – A3 (written)

1) Machine Learning & Neural Networks

(a) Adam Optimizer

i. The first moment estimate, also known as the momentum term, is an **exponentially decaying average of past gradients**. It helps to smooth out the updates and reduce the impact of **noisy gradients**, leading to a **more stable convergence** during training. The impact of the momentum term depends on the value of beta1. A high value of beta1 means that the momentum term will give more weight to the recent gradients, resulting in a smoother update process that adapts quickly to changes in the gradient. On the other hand, a low value of beta1 means that the momentum term will give less weight to the recent gradients, resulting in a slower update process that is more resistant to noisy or fluctuating gradients.

ii. In the Adam optimizer, **the model parameters that have large gradients and small historical momentum values will get larger updates.** This is because the Adam optimizer scales the step size for each parameter based on the magnitude of its gradient, with larger gradients resulting in smaller step sizes and vice versa.
This approach can help with learning because it allows the optimizer to make larger updates to the parameters that are most important for reducing the loss function, while making smaller updates to those that are less important or more susceptible to noise. By **adapting the step size** for each parameter based on the current gradient, Adam can more effectively navigate complex loss landscapes and converge to optimal solutions more quickly than traditional gradient descent methods.

(b) Dropout

i. In dropout, γ represents the scaling factor applied to the activations of the remaining neurons after dropout. It's commonly used to ensure that the expected value of the output from the layer remains the same as if no dropout was applied.

The formula for calculating the scaling factor γ is:

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

where p_{drop} is the probability of dropping out a neuron during training. Because the expected value of h_{drop} is still h and the expected value of d is $(1 - p_{drop})$.

ii. Dropout is a regularization technique used **during training** to prevent **overfitting**. It works by randomly "dropping out" (setting to zero) some of the neurons in a neural network during each forward pass. By doing so, dropout forces the remaining neurons to learn more robust representations of the data, since they cannot rely on the presence of any one specific neuron.
During evaluation or testing, however, we want the neural network to use all of its neurons and make predictions based on the full strength of the model. Therefore, we should not apply dropout during evaluation. If we did, we would effectively be taking an average of many different sub-networks, which would **not accurately represent the predictions of the full**, trained model.

2) Neural Transition-Based Dependency Parsing

(a)

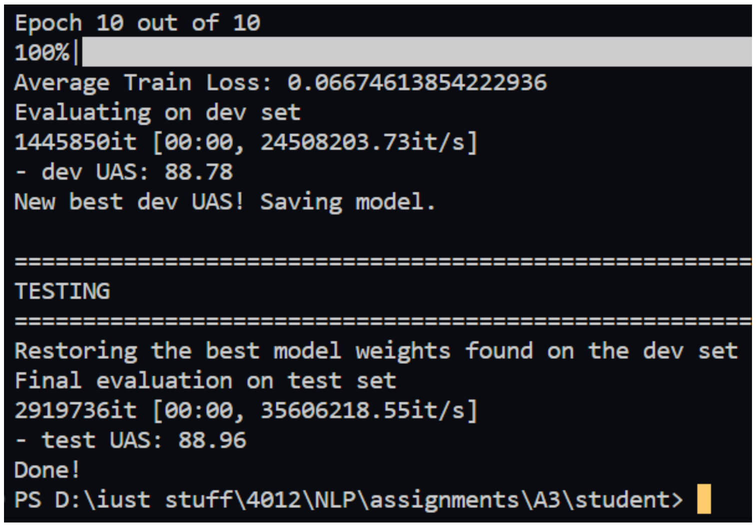
Stack	Buffer	New dependency	Transition
[ROOT]	[I, attended, lectures, in, the, NLP, class]		Initial configuration
[ROOT, I]	[attended, lectures, in, the, NLP, class]		SHIFT
[ROOT, I, attended]	[lectures, in, the, NLP, class]		SHIFT
[ROOT, attended]	[lectures, in, the, NLP, class]	attended → I	LEFT-ARC
[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)

2n times. Because first we add all words (n) to Stack from the Buffer, then we remove them (n) from the Stack.

(e)



```
Epoch 10 out of 10
100%|
Average Train Loss: 0.06674613854222936
Evaluating on dev set
1445850it [00:00, 24508203.73it/s]
- dev UAS: 88.78
New best dev UAS! Saving model.

========================================================
TESTING
========================================================
Restoring the best model weights found on the dev set
Final evaluation on test set
2919736it [00:00, 35606218.55it/s]
- test UAS: 88.96
Done!
PS D:\iust stuff\4012\NLP\assignments\A3\student>
```

(f)

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 → decision
Correct dependency: reasons → decision

iv. **Error type**: Coordination Attachment Error
Incorrect dependency: affects → one
Correct dependency: plants → one

(g)

Part-of-speech (POS) tagging is the process of assigning grammatical tags to each word in a sentence, such as noun, verb, adjective, adverb, etc. These tags can be very useful features in a parser for several reasons:

Syntactic disambiguation: Many words in natural language can have multiple meanings and functions depending on their context. For example, the word "bank" can be a noun referring to a financial institution or a verb meaning to deposit money. By using POS tags as features, a parser can disambiguate the syntactic role of a word based on its part of speech, which helps to correctly assign a parse tree.

Contextual information: POS tags provide contextual information about a word's neighboring words, which can help to inform the parser about the structure of the sentence. For example, if a verb like "run" is preceded by an article like "the", it is more likely to be a transitive verb with an object following it.

Reduced ambiguity: By providing more information about the words in a sentence, POS tags can reduce the overall ambiguity of the parsing task. This can lead to improved accuracy and efficiency of the parser, particularly when dealing with complex or ambiguous sentences.