NLP Assignment 3

Analysis (Experiments)

Before testing the experiments as mentioned in the assignment paper, I ran my model on the default configurations given and got the below scores:

Experiments	UAS	UASno Punc	LAS	LASno Punc	UEM	UEMno Punc	Root	Loss
Default Configuration	70.5361816686	73.3086531397	66.3708652192	68.7164415305	10.8235294118	11.3529411765	64.4705882353	0.3614805272221565

Then I tried increasing the learning rate and number of iterations to see how the model performs. I saw a significant increase in the performance of the model at iterations 2001 and learning rate 0.3, so I fixed these two hyperparameters and performed the below testing.

1. **Number of hidden layers** – As per the paper, only one hidden layer was implemented. Here we had to implement 2 more hidden layers.

   Configuration: Default with learning rate 0.3, hidden size=150 and number of iterations 2001

Experiments	UAS	UASno Punc	LAS	LASno Punc	UEM	UEMno Punc	Root	Loss
Hidden Layer	80.0882418925	82.1652630984	76.9274871002	78.5932289606	20.8235294118	22.2352941176	82.0588235294	0.20614504769444467
Hidden Layer 1	81.2672931675	83.1967444752	78.3533165491	79.9129599277	22.5882352941	24.1764705882	82.4705882353	0.20979091927409171

Hidden Layer 2	18.45	19.50	14.67	1.56	0.64	0.645	14.877	1.133

Observation: As we can see from the result, increase in hidden layer doesn't necessarily increase the accuracy a lot. By adding one hidden layer of 150 nodes, I saw an increase in the scores, but when I added a third layer of 100 nodes I saw a considerable decrease in the performance. The loss computed was very high as evident from the scores above. Also, in the paper it was mentioned that adding more and more hidden layers do not necessarily have an impact in the performance and that is quite evident from the results.

2. **<u>Capturing interactions</u>** – The paper used a cube non-linearity to account for interactions between different combinations of token features. We had to implement sigmoid, relu and tanh to see the performance.

   Configuration: Default with learning rate 0.3 and number of iterations 2001

Experiments	UAS	UASnoPunc	LAS	LASnoPunc	UEM	UEMnoPunc	Root	Loss
Cube Activation	80.0882418925	82.1652630984	76.9274871002	78.5932289606	20.8235294118	22.2352941176	82.0588235294	0.20614504769444467
Relu	78.9042051998	81.0772621941	75.6562055986	77.4317526705	18.7647058824	19.8235294118	75.1764705882	0.2533218115568161
Sigmoid	68.5619562779	71.4972022834	62.8187551412	65.2800542587	9.0	9.52941176471	60.6470588235	0.4785803160071373
Tanh	78.2087394371	80.4442434861	74.831118977	76.643305262	17.2352941176	18.2352941176	76.0588235294	0.2651348215341568

Observation: Clearly as the paper suggest the cube activation function outperforms the other functions.

3. **<u>Parallel Layers each for word, Pos tag and deps</u>**–

Configuration: Default with learning rate 0.3 and number of iterations 2001

Experiments	UAS	UASnoPunc	LAS	LASnoPunc	UEM	UEMnoPunc	Root	Loss
Parallel hidden layers	80.6790138844	82.7982818064	77.6877632924	79.4240660148	21	22.2352941176	79.7647058824	0.238249968290329

Observation: By implementing separate hidden layers, the performance of the model did not improve. This might be because there is no interaction between the words, pos tags and labels and hence the decrease in scores.

4. **Effect of fixing Word, POS and Dep Embedding's** – Fix word embeddings and using the pre-trained model.

Configuration: Default with learning rate 0.3, hidden size=150 and number of iterations 2000

Experiments	UAS	UASnoPunc	LAS	LASnoPunc	UEM	UEMnoPunc	Root	Loss
Fixed Embeddings	40.8	48.731701803	34.5	37.8	2.87	2.90	33.5	0.65

Observation: I noticed a remarkable decrease in the in the scores by setting trainable as False. This is because the model is not learning new interactions and therefore the performance is not good.

5. **Best Configuration –**

I tried to increase the interactions to 4000 with learning rate as 0.3 to see how well the model works as with lesser iterations the loss value was not converging properly. The model accuracy increased significantly by doing so and the loss value also converged.

Parameter	Value
Maximum Iterations	**4001**
Batch Size	10000
Hidden Size	200
Embedding Size	50
Learning Rate	0.3
Display Step	100

Lambda Value	10^{-8}
Validation Step	200
Activation Function	Cube Activation
Number of tokens	48

Experiments	UAS	UASnoPunc	LAS	LASnoPunc	UEM	UEMnoPunc	Root	Loss
Best	84.624971957	86.4663991409	82.0724381185	83.5867292149	27.4117647059	29.1176470588	84.4117647059	0.16356452628970147

6. **<u>Gradient Clipping</u> -** Gradient Clipping: It is a method which clips the gradients or caps them to a threshold value to prevent it from getting too large or too small.

Usefulness: Gradient Clipping is common in recurrent neural networks where gradients are being propagated both in forward and backward mode.

There are two problems to look into:

1. Vanishing gradient Issue: It is seen that when the gradient of the loss function is multiplied with numbers less than 1 then the gradients tend to become vanishingly small and hence the name vanishing gradient problem.
2. Exploding gradient Issue: Sometimes the other way happens. The gradient of the loss function when multiplied with numbers greater than 1 then there is a possibility that the gradient explodes and hence the name exploding gradient problem.

In both the cases, we need to set a threshold so that the gradients do not become too large or too small. Hence, we use gradient clipping to clip the numbers to prevent them from the above two problems.

By removing gradient clipping:

Configuration: Default with learning rate 0.1 and number of iterations 1001

Experiments	UAS	UASnoPunc	LAS	LASnoPunc	UEM	UEMnoPunc	Root	Loss
Parallel hidden layers	nan	nan	nan	nan	nan	nan	nan	nan

Observation: The loss values after the first one (started with 4.51 at step 0) started to come as Nan after removing the gradient clipping.