NLP with Deep Learning
Mini Project 2

Name: Poojith Reddy Maligireddy
UID: u1405749

1)I have implemented the model and the parser in the file named main.py and also filled state.py and submitted the results.txt in a specified format.

2)Below are the results generated (added best learning rate column as well):

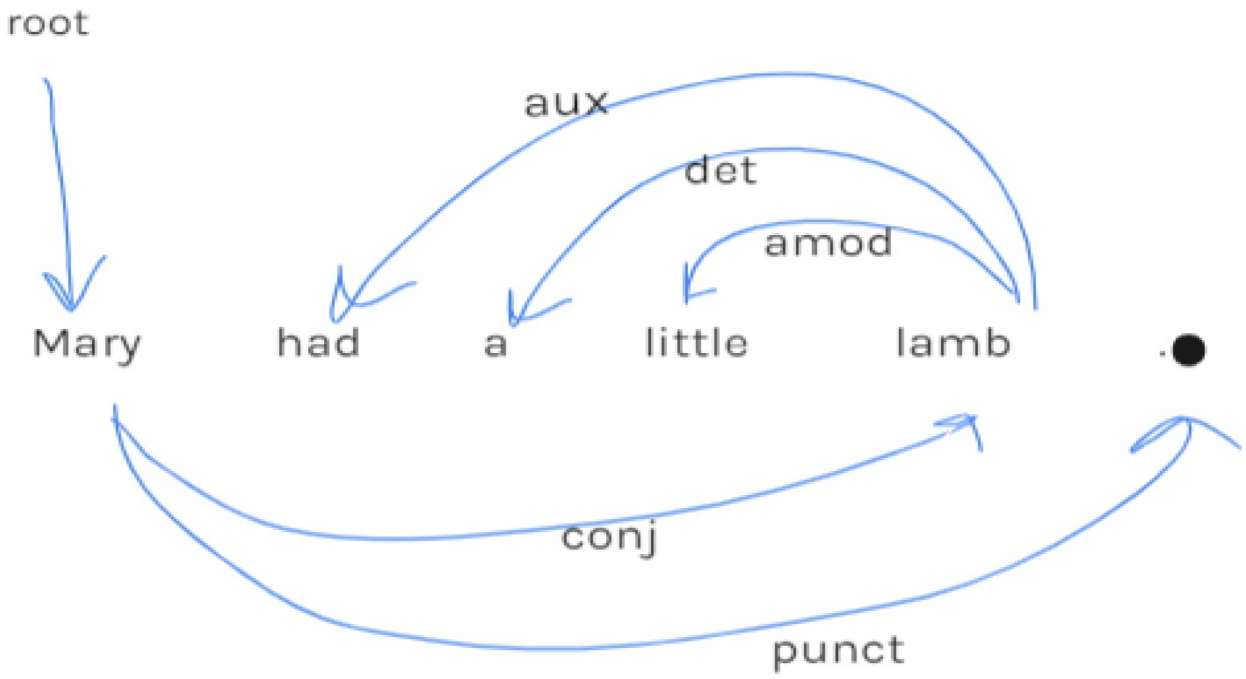
Embedding	**Best LR**	**Mean**		**Concatenate**	
		UAS	**LAS**	**UAS**	**LAS**
Glove 6B 50d	0.001	40.23	35.32	65.43	60.13
Glove 6B 300d	0.0001	41.76	36.87	65.87	61.29
Glove 42B 300d	0.0001	42.29	38.45	66.21	63.45
Glove 840B 300d	0.0001	44.86	40.37	69.27	66.67

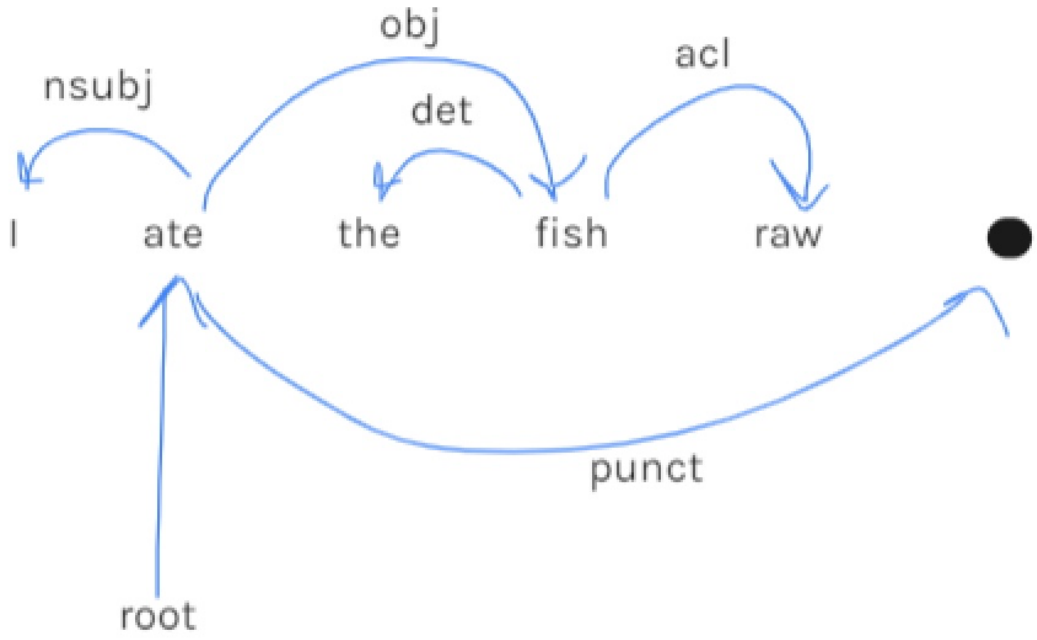
3) In above results, we could see that UAS score are higher than LAS scores and they are as expected because in LAS score we are checking whether our models is predicting correct label in addition to what we are checking in UAS score.

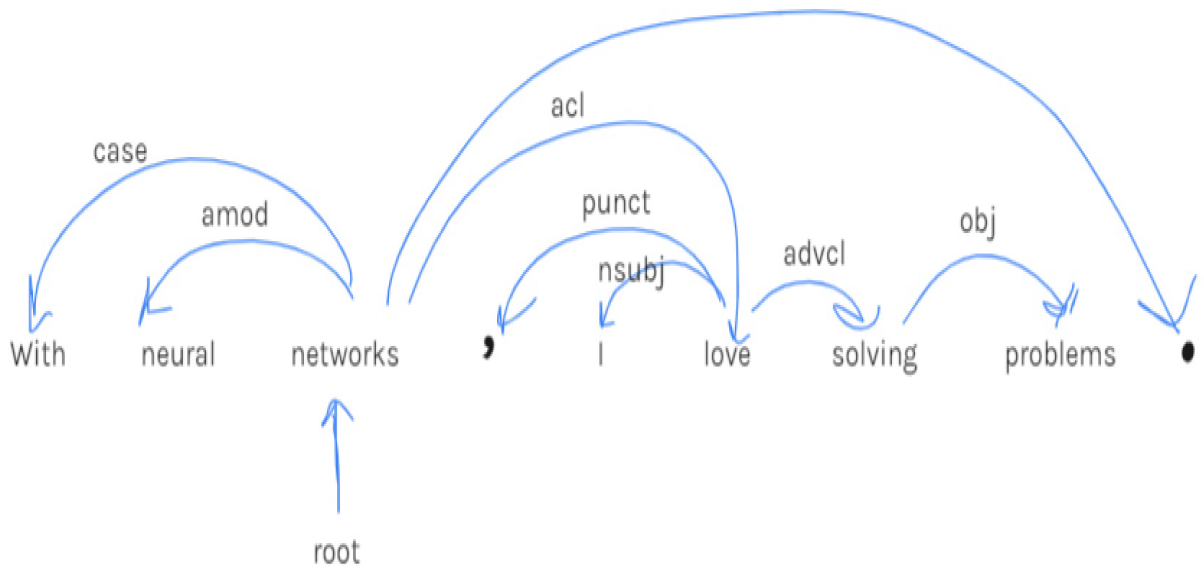
Also, we are getting good scores in concatenation than mean and they are as expected because sentences are sequence of words and we are somewhat capturing some sequence information while concatenating and this is not in the case of mean.

Finally, accuracies are increasing with word embedding size and it is true because the more the size of word embedding the more it captures the information about that word.

4)Below are the dependency parse trees:







5) The paper's approach differs from ours in following ways:

We have used two words each from stack and buffer and their corresponding POS tags but in paper, in addition to that, it uses dependency arc labels, an extra feature we partially implemented in the extra credit question but not in the primary one(more details in next paragraph). The paper utilizes POS and label embeddings and expect these semantic meanings to be effectively captured by the dense representations and uses a cube activation function where we used ReLU activation function in our implementation.

In our implementation, we used top two elements from stack and first two elements from buffer and so total number is four for word embeddings and the four corresponding POS tags but paper uses total of eighteen words for word embeddings(top three each from stack and buffer and the first and second leftmost / rightmost children of the top two words on the stack and the leftmost of leftmost / rightmost of rightmost children of the top two words on the stack) and their respective eighteen POS tags and also corresponding arc labels of words excluding those six words on the stack/buffer i.e; a total of twelve corresponding arc labels. The good advantage of our parser is that we can add a rich set of elements cheaply, instead of hand-crafting many more indicator features.

In paper, they generated training examples from training sentences and their gold parse trees by adopting a shortest stack oracle strategy which always prefer left-arc

operations over shift operation. In our implementation, we used cross entropy loss but in paper, they used L2 regularization in addition to the cross-entropy loss which further refines the model and they used pre-trained word-embeddings. Also, we have used Adam optimizer for our implementation but they have used mini-batched AdaGrad and applied dropout with 0.5 rate which is different from our implementation. In addition to that, we have used best LAS score on development set for final evaluation but in paper, they used best UAS score for final evaluation. Finally, in paper, they applied a pre-computation trick to speed up the parsing time.

Extra Credit:

I have implemented the model and the parser in the file named main_extracredit.py. In this file I have created few extra functions to find the left most and right most child for top two elements in the stack and corresponding dependency arc labels to compute their embeddings and then trained.

I am reporting results for, Glove 6B 300d with concatenation for word embedding and mean for POS tags and dependency arc labels and got the highest test UAS score of 67.34 and LAS score of 62.45 with best learning rate of 0.0001.

So, we could see the increase in accuracies with using dependency arc label embeddings and left and right most child word embeddings of top two stack elements. There is more increase in accuracy if we use concatenation for both POS tags and dependency arc labels as well or using higher parameter word embeddings with concatenating word embeddings and also POS tags and dependency arc label embeddings.