1.
Random seed used 42

1 Implemented model in the model.py, model is parameterized, therefore same model class for all mean, concat, and with label.
2 Implemented parser in the dependency_parser.py It contains two parser, one is directParser - which is used for generating the training dataset, other is actualParser - which is used during the evaluation loop for getting the list of predicted action for the sentence.
3 results.txt contains the space separated predicted actions, NOTE - one extra line in the end, following the similar pattern in other txt files.

2.
Results - **without label** on the test dataset, trained using training and best model picked for best LAS on dev dataset.

Embedding	Mean		Concatenate	
	UAS	LAS	UAS	LAS
GloVe 6B 50d	0.396372518	0.33795863215	0.68438831722	0.61979004261
GloVe 6B 300d	0.4242282507	0.36981602743	0.67721650556	0.61526868308
Glove 42B 300d	0.4457956553	0.38587464920	0.68106225964	0.61942625506
GloVe 840B 300d	0.4416900530	0.38629040640	0.70408481446	0.64244880989

Stored in /scratch/general/vast/u1471428/cs6957/assignment2/models/

Model1 - GloVe 6B 50d - with_mean, epoch 6, learning_rate - 0.001
Model2 - GloVe 6B 300d - with_mean, epoch 8, learning_rate - 0.001
Model3 - GloVe 42B 300d - with_mean, epoch 10, learning_rate - 0.001
Model4 - GloVe 840B 300d - with_mean, epoch 8, learning_rate - 0.001

Stored in /scratch/general/vast/u1471428/cs6957/assignment2/models/concat

Model5 - GloVe 6B 50d - with_concat, epoch 7, learning_rate - 0.0001
Model6 - GloVe 6B 300d - with_concat, epoch 6, learning_rate - 0.001
Model7 - GloVe 42B 300d - with_concat, epoch 18, learning_rate - 0.0001
Model8 - GloVe 840B 300d - with_concat, epoch 12, learning_rate - 0.0001

Best Model is **Model8 - GloVe 840B 300d - with_concat, epoch 12, learning_rate - 0.0001**

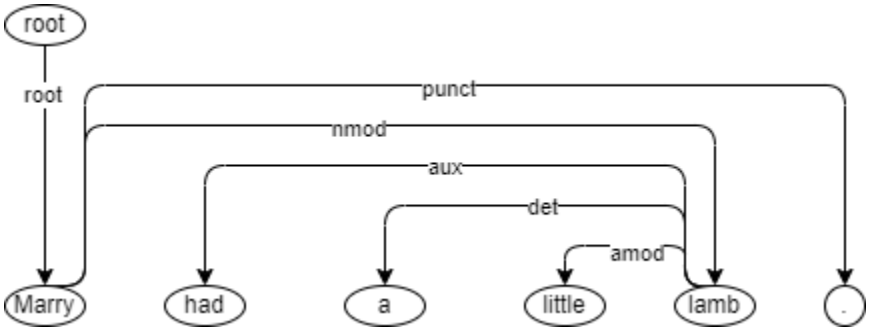
3.

Trends

- The UAS score is always greater than the LAS score, as LAS also considers the dependency relation, while UAS only considers the correct head.
- Score increases, with increase in the token used in GloVe embeddings.
- However, the above increase in score is not that significant, as compared to mean vs concatenation
- For mean, the score increases as the dimension increases, but this is not observed in the concatenation part.
- Score for the concatenation is very superior as compared to the score obtained with mean. This could be attributed to the fact that during concatenation, the model has more space to learn.
- (BONUS), with the addition of label, leftmost and rightmost word of top-2 stack tokens, the score reaches above 0.75 in concat model, which could be attributed to the fact that the model has more dimensionality word representation.

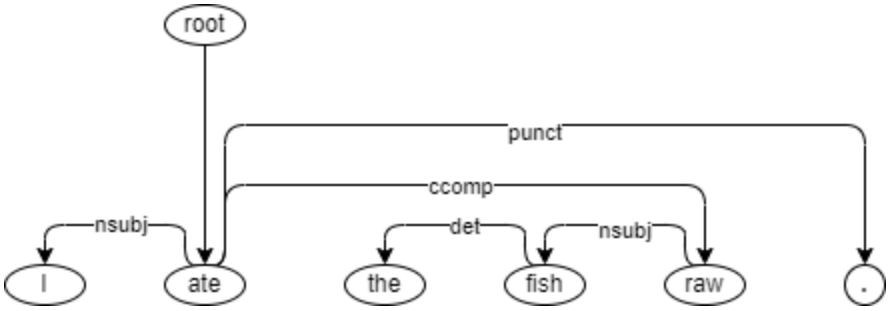
4.

Sentence - POS Mary had a little lamb . ||| PROPN AUX DET ADJ NOUN PUNCT
Prediction - SHIFT SHIFT SHIFT SHIFT SHIFT REDUCE_L_amod REDUCE_L_det REDUCE_L_aux REDUCE_R_nmod SHIFT REDUCE_R_punct



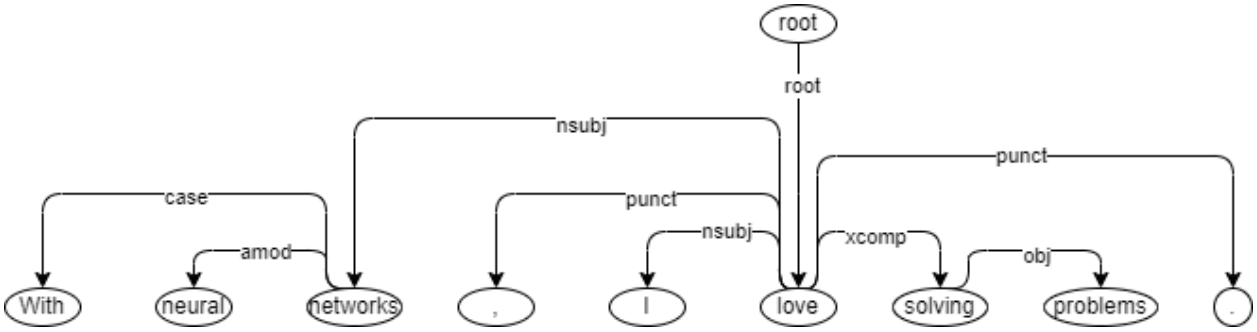
Sentence - POS I ate the fish raw . ||| PRON VERB DET NOUN ADJ PUNCT

Prediction - SHIFT SHIFT REDUCE_L_nsubj SHIFT SHIFT REDUCE_L_det SHIFT REDUCE_L_nsubj REDUCE_R_ccomp SHIFT REDUCE_R_punct



Sentence - POS With neural networks , I love solving problems . ||| ADP ADJ NOUN PUNCT PRON VERB VERB NOUN PUNCT

Prediction - SHIFT SHIFT SHIFT REDUCE_L_amod REDUCE_L_case SHIFT SHIFT SHIFT REDUCE_L_nsubj REDUCE_L_punct REDUCE_L_nsubj SHIFT SHIFT REDUCE_R_obj REDUCE_R_xcomp SHIFT REDUCE_R_punct



5.

Difference between our representation of parse state and representation used in paper -

- Activation function - we have used ReLU as the activation function, but in the paper, authors have used **cubic activation function.**

  // below are the difference between parse_state
- **In the normal implementation, we are only considering the word and their POS tags, while in the paper, authors have also considered labels between the words in the input during the training.**
- **In our implementation, we only use 4 words representation 2 from each stack and buffer, while in the original paper they use 18 words representation**
  - **3 from each stack and buffer, //** adding 6 words
  - **1st and 2nd leftmost and rightmost child of top 2 stack elements,** adding 8 words // NULL is used for non-existent element
  - **leftmost of leftmost and rightmost or rightmost for top 2 elements of stack //** adding 4 elements // NULL is used for non-existent elements.

- **In our implementation, we only use 4 pos-tag, while in the original paper they use 18 corresponding POS tags corresponding to all words. //** (in the bonus part, when we add the leftmost and rightmost words, but not their pos-tag).
- **(BONUS) we add the dependency labels of leftmost and rightmost childs of top 2 stack elements, while in the original paper they add 12 labels. //** corresponding to the label arc for 8 from (1st and 2nd leftmost and rightmost of top 2 stack words) and 4 from (leftmost of leftmost and rightmost of rightmost).

  The original paper uses a very rich set of elements instead of handcrafted features, we also did similarly but very less set of elements.

- For generating the training input, they have parsed using a shortest stack oracle which always prefers Left-arc over the shift, in our case shift seems to have priority. (based on the provided data observation).

BONUS

Results - **with label** on the test dataset, trained using training and best model picked for best LAS on dev dataset.

Embedding	Mean		Concatenate	
	UAS	LAS	UAS	LAS
GloVe 6B 50d	0.3935661573	0.33276166718	0.78385822679	0.75350795135
GloVe 6B 300d	0.4047916017	0.34882028895	0.78385822679	0.75210477081
Glove 42B 300d	0.4163808335	0.36176073173	0.78271489450	0.75111734746

Model1 - GloVe 6B 50d - with_mean, at epoch 6 with learning rate 0.001
Model2 - GloVe 6B 300d - with_mean, at epoch 10 with learning rate 0.001
Model3 - GloVe 42B 300d - with_mean, at epoch 3 with learning rate 0.001
Model5 - GloVe 6B 50d - with_concat, at epoch 8 with learning rate 0.001
Model6 - GloVe 6B 300d - with_concat, at epoch 8 with learning rate 0.001
Model7 - GloVe 42B 300d - with_concat, at epoch 4 with learning rate 0.001