Implementation Details -

`Model.py -`

1. The cubic activation was straightforward to execute: the input tensor was raised to the power of three and returned.
2. The model layer weights were randomly initialized in the init method. Some of the important variables which were initialized are -
    a. Embeddings - dims ([vocab_size, embedding_dim]) and standard dev of 0.35
    b. Hidden_weights - dims([num_tokens*embedding_dim, hidden_dim]) and standard dev of 0.02
    c. Hidden_bias - initialized to zeros
    d. Output_weights - dims([num_transitions, hidden_dim]) and standard dev of 0.02
3. In the call method, the inputs were processed as mentioned in the paper. And the logits were returned.
4. In the compute_loss method, (-1) labels were used to create masks and softmax was computed. Later only labels with value (1) were isolated then the final loss was computed.
5. Later the regularization term was computed and added to the loss and the combined sum was returned.

`Parsing_system.py -`

1. The apply function was implemented as mentioned in the paper.
2. At first two operands are popped and I checked if one of the three operations were applicable and executed if they were applicable.
3. The transformed configuration object was returned.

`Data.py -`

1. The get_configuration_features method was implemented as described in the paper - "A Fast and Accurate Dependency Parser using Neural Networks"(2014)
2. The features, postags and arclabels were collected by parsing the trees formed in the previous steps. The nodes of the trees were collected as labels.
3. After getting the features, postags and arclables their respective ids were also fetched and appended to a list.
4. The final list created above was returned.

Results on dev set -

Following are the results of various models created by using different activation functions. I got the best results based on cubic activation function. So, I produced the final predictions on the model trained by using cubic activation. My findings for other activation functions are as below.

1. Cubic activation gives the best possible UAS.
2. Model trained without embeddings, model trained on tanh activation and model trained without GloVe are comparable.
3. The model trained on sigmoid is the least preferable model.

Cubic Activation UAS: 87.81065383752524 UASnoPunc: 89.3601989487368 LAS: 85.29301792257647 LASnoPunc: 86.51444073927541 UEM: 34.529411764705884 UEMnoPunc: 37.23529411764706 ROOT: 90.58823529411765	Sigmoid UAS: 85.48246379340429 UASnoPunc: 87.36788560447634 LAS: 82.9673205872822 LASnoPunc: 84.5164754422653 UEM: 29.235294117647058 UEMnoPunc: 31.705882352941178 ROOT: 84.11764705882354
Tanh UAS: 87.26225789565521 UASnoPunc: 88.9476063980105 LAS: 84.78450532193334 LASnoPunc: 86.14141185779687 UEM: 32.64705882352941 UEMnoPunc: 35.294117647058826 ROOT: 87.05882352941177	Without_emb_tune UAS: 87.80065383752524 UASnoPunc: 89.26019374487368 LAS: 85.09301792257647 LASnoPunc: 86.31349073927541 UEM: 34.220417164705884 UEMnoPunc: 37.13529401764706 ROOT: 90.38023529411765
Without Glove UAS: 87.63367151083082 UASnoPunc: 89.18216243712203 LAS: 85.21075853129597 LASnoPunc: 86.452269259029 UEM: 35.23529411764706 UEMnoPunc: 38.11764705882353 ROOT: 89.11764705882354	