Aditya Choudhary

NLP Hw-3 Report

1. Model Implementation

a. The arc-standard algorithm

- .First I checked if the transition is Left-Arc or Right-Arc, if yes extracted out ‘label’.
- Then as per the paper if ‘Left’ then add an arc from Stack[0] to Stack[1] and pop Stack[1]. Or if ‘Right’ then add an arc from Stack[1] to Stack[0] and pop Stack[0].
- If transition is Shift , moved Buffer top to Stack by using ‘shift’ function of configuration object.

b. Feature extraction

- I followed the exact sequence of nodes(stack-tops,buffer-tops , left-childs, right-childs, left-of-left, right-of-right) to be extracted as suggested in the paper("A Fast and Accurate Dependency Parser using Neural Networks"[2014]).
- Configuration object already has all the required functions pre-written to extract out the corresponding elements.
- I generated a list of 18 ids as suggested in the paper and then processed it to get corresponding words(18), pos_tags(18) and labels(12) from the tree.
- Then these 48 words were converted to vocabulary index and returned as one features list.

c. the neural network architecture including activation function

- For the architecture, First I initialized weights for the two layers using tf.Variable and tf.random.truncated_normal methods. Dimension of w_1 is [(embedding_dim * num_tokens) X hidden_dim] and dimension of w_2 is [hidden_dim X num_transitions] .
- Bias weights b_1 is initialized to Zero , dimension = [hidden_dim]
- Embeddings are initialized for dimension [vocab_size X embedding_dim] and is selected from random-uniform distribution in range (-0.01 , 0.01) as suggested in the Research paper. Embeddings are trainable only if ‘trainable_embeddings’ is passed in __init__ method.
- In the forward pass, I first did embedding_lookup for each word in a sentence and then reshaped it to ‘concatenate’ all words embeddings in a sentence. This results in single vector per sentence.
- Then performed the operation - W_1 * emb_inputs + b_1
- Applied activation function on the above output, multiplied with second weights matrix w_2 to get ‘logits’.
- These ‘logits’ is used to calculate batch_loss.

- For Cubic activation function I performed element-wise cube to the vector and return it.

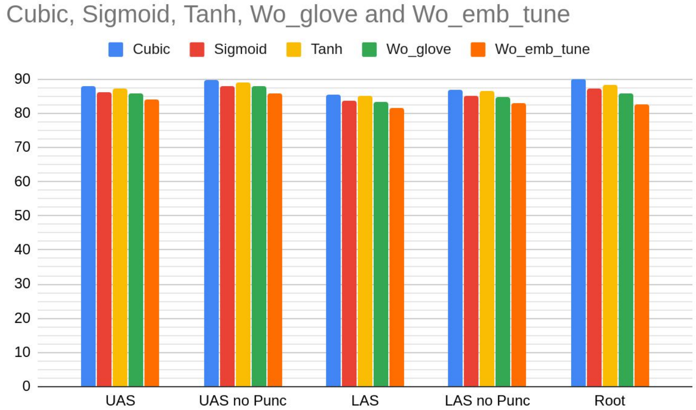
2. loss function

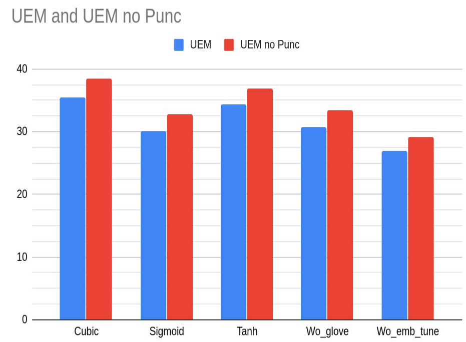
- we compute the softmax probabilities only among the feasible transitions (label = 0 and 1). Thus used labels to mask out the infeasible moves.
- For calculating cross-entropy loss only correct labels need to be used. Thus used label = 1 to mask correct labels among the feasible ones.
- I used a small no. epsilon 1e-10 to prevent any edge case causing log(0) resulting in ‘nan’.
- In regularization calculation, embeddings l2-loss is calculated only when trainable_embeddings is set true. In all other cases Regularization term is calculated with both weight matrices, and bias vector.

3. Results and Analysis

3.1 Results Table

	UAS	**UAS no Punc**	**LAS**	**LAS no Punc**	**UEM**	**UEM no Punc**	**Root**
Cubic	87.972	89.642	85.405	86.734	35.411	38.529	89.941
Sigmoid	86.113	87.896	83.647	85.104	30.058	32.705	87.352
Tanh	87.424	89.043	85.086	86.395	34.294	36.882	88.176
Wo_glove	85.968	87.794	83.418	84.917	30.764	33.352	85.823
Wo_emb_tune	84.049	85.881	81.451	82.959	27.0	29.176	82.588





- From the above table and figures we can infer that Cubic activation function outperforms the other two. Performance of the activation functions for Dependency parsing task can be written as - **Cubic > Tanh > Sigmoid.**
- Cubic function gives 0.5 - 1.8% improvement over tanh and sigmoid function in UAS metrics.
- Pretained embeddings gives around 2% improvement over training embeddings from scratch in UAS and LAS metrics.
- This is coherent with what was analysed in the research paper.
- When we trained embeddings from scratch it achieves comparable accuracy across all the metrics.
- Accuracy decreasses in every metrics when embeddings are kept frozen. This is because the model is not allowed to learn the word relations and context, specific to this dataset. Glove embeddings are trained on different task and dataset and generally cannot represent(to 100 %), every type of downstream dataset it is used for.