Stony Brook University
Department Of Computer Science

CSE 538 Natural Language Processing - Fall 2019
Instructor: Niranjan Balasubramanian

Assignment 3

Madhusmita Dash (SBU ID: **112715430**)

1. Model Implementation:

The arc-standard algorithm:
There are three components maintained inside the configuration object: stack, buffer and dependency graph constructed till now. The following operations are carried out:

Shift: If buffer is not empty, shift the word from the buffer to the stack and remove it from buffer.
Left Arc: Get the top two elements from stack, draw an arc (with corresponding label) from top of stack to the next top element of stack, remove the second top element from the stack.
Right Arc: Get the top two elements from stack, draw an arc (with corresponding label) from second top of stack to the top element of stack, remove the top element from the stack.

Feature extraction:

We calculate the features by considering the words, POS tags of words and the labels of the arcs. There will be 18 features for the words in the stack, 18 features for their POS and 12 features for the labels.

We get the top 3 elements from the stack and the first three elements from the buffer.
We fetch the word id/pos id/label id of the word/pos/tag from the vocabulary.

Features from words (18):
Take the top 3 stack elements s1,s2,s3 (3)
First 3 buffer elements b1,b2,b3 (3)
First and second Left child, right child of first two stack elements s1,s2 (8)
Left child of left child, right child of right child of s1,s2 (4)

Features from pos (18):
Take the pos of top 3 stack elements s1,s2,s3 (3)
First pos of 3 buffer elements b1,b2,b3 (3)
Pos of first and second Left child, right child of first two stack elements s1,s2 (8)
Pos of left child of left child, right child of right child of s1,s2 (4)

Features from words (12):
Label of first and second Left child, right child of first two stack elements s1,s2 (8)
Label of left child of left child, right child of right child of s1,s2 (4)

Model and loss function:

Initialization:
Embeddings are initialized randomly uniformly in the range (-0.01,0.01), bias with zeros and weights with truncated_normal from a distribution with mean=0.0 and standard deviation of sqrt(2/(dim0+dim1))
Embeddings: [vocab_size, embedding_dim] with trainable set by the function parameter
Weight of first hidden layer: [hidden_dim, self._embedding_dim * self._num_tokens]
Bias of first hidden layer: [1, hidden_dim]
Weight of output layer: [num_transitions, hidden_dim]

Model:
First we fetch the embeddings of the inputs and calculate hidden layer output h as activation(input_emb*W_layer1 + bias). The activation can be cubic, sigmoid or tanh.
Then we calculate the logits by taking h*W_layer2 and get the loss from compute_loss function.

Softmax and Cross entropy loss:
In compute loss, we first filter out feasible transitions in which labels have 0 or 1 entries and take the exp of such values (denominator of softmax). For the correct labels i.e 1 we calculate the numerator by applying the mask for correct labels and find out the softmax value by dividing the above numerator by denominator.
We calculate the loss by taking the log of the softmax values of the correct labels and find the L2 regularization term to penalize the updates of the training parameters of the model.

2. Experiments:

a. Activations (cubic vs tanh vs sigmoid)

Activation	Results
Cubic	**UAS: 87.85**053717875215 UASnoPunc: 89.39128468886 LAS: 85.34536480793679 LASnoPunc: 86.5511784321483 UEM: 34.294117647058826 UEMnoPunc: 37.05882352941177 ROOT: 90.6470588235294

Tanh	UAS: 87.40683500760277 UASnoPunc: 89.06347142937885 LAS: 85.06618141934841 LASnoPunc: 86.4155315661561 UEM: 34.05882352941177 UEMnoPunc: 36.88235294117647 ROOT: 88.17647058823529
Sigmoid	UAS: 86.11311912655482 UASnoPunc: 87.92742892669418 LAS: 83.6303811351796 LASnoPunc: 85.11558243373085 UEM: 30.705882352941178 UEMnoPunc: 32.88235294117647 ROOT: 87.58823529411765

Observation: The cubic activation as mentioned in the paper gives best results as compared to the sigmoid and tanh activations. Tanh performs better than sigmoid activation. Using the cubic activation function better captures the interaction of three elements which is not included in the other activation functions like tanh and sigmoid thus giving better performance.

b. Pretrained embeddings

Pretrained embeddings	**Results**
False	UAS: 83.6303811351796 UASnoPunc: 85.65534392132481 LAS: 81.12022334671087 LASnoPunc: 82.85197535748601 UEM: 26.705882352941178 UEMnoPunc: 29.0 ROOT: 78.70588235294117

Observation: As compared to the above models where we were using the pre-trained glove embeddings, when we use random initialization of the embeddings, we get relatively less UAS. This is because if we use an already pretrained embeddings, then the model is already aware of some meaningful representation than starting with random initialization.

c. Tunability of embeddings

Embeddings tunable	**Results**
False	UAS: 84.85679387790712 UASnoPunc: 86.67269541626632 LAS: 82.03504748610315 LASnoPunc: 83.49347199457412 UEM: 28.235294117647058 UEMnoPunc: 30.352941176470587 ROOT: 85.0

Observation: Without using tunable embeddings, we get lesser UAS than if we use tunable embeddings. This is because the POS tags add semantic meaning to the representation of the words thus having better performance.

References:

https://nlp.stanford.edu/pubs/emnlp2014-depparser.pdf
https://pdfs.semanticscholar.org/0329/06f0d86fd6bbf7512d3fffd06fcb83593012.pdf