CS224n Winter 2019 Homework 4

SUNet ID: 05794739
Name: Luis Perez
Collaborators:

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Problem 1

(a) The embedding must capture the essence of the item being embedded, and this is reflected by the dimensionality of the embedding – in other words, the higher the information content (number of unique items), the larger the embedding dimesion should be. Considering that a typical vocabulary size of words, $|\mathcal{V}|$, can be in the thousands or hundreds of thousands (with the number of possible words far higher), while the size of the character set for most languages, $|\mathcal{C}|$, is typically a few orders of magnitude smaller (in the hundreds), it is reasonable that a character embedding of 50 suffices.

(b) The number of parameters for the word-based lookup embedding model is trivial to compute (where we treat the embedding itself as trainable). We have the number of parameters as:

$$V_{\text{word}} \times e_{\text{word}} = 12.8M$$

The number of parameters for the character-based embedding model is a little more involved to compute, but can nonetheless still be done. We have:

$$\begin{aligned} V_{\text{char}} \times e_{\text{char}} &= 4,800 && \text{(Character Embedding Parameters)} \\ e_{\text{word}} \times e_{\text{char}} \times k + e_{\text{word}} &= 64,256 && \text{(Convolution Parameters)} \\ 2 \times [e_{\text{word}} \times e_{\text{word}} + e_{\text{word}}] &= 131,584 && \text{(Highway Network Parameters)} \end{aligned}$$

This gives a final expression for the number of parameters as:

$$V_{\text{char}} \times e_{\text{char}} + e_{\text{word}} \times e_{\text{char}} \times k + e_{\text{word}} + 2 \times [e_{\text{word}} \times e_{\text{word}} + e_{\text{word}}] = 200,640$$

From the above calculations, it is clear that the word-embedding model has more parameters, by a factor of 64 (almost two orders of magnitude).

Problem 2

(a) In “char_decoder.py”.

(b) In “char_decoder.py”.

(c) In "char_decoder.py".

(d) In "char_decoder.py"

(e) Results in "outputs/tesr_outputr_locar_q2.txt"

(f) Training took 40611.18 sec (11.28 hours) with a final test BLEU score of 24.56565986092025. Sampled results can be found in "outputs/tesr_outputs.txt".