Author: Benjamin Smidt
Created: October 31st, 2022
Last Updated: November 2nd, 2022

CS 224N A4: Neural Machine Translation with RNNs

Note to the reader. This is my work for assignment 4 of Stanford's course CS 224N: Natural Language Processing with Deep Learning. You can find the lecture Winter 2021 lectures series on YouTube here. This document is meant to be used as a reference, explanation, and resource for the assignment, not necessarily a comprehensive overview of Word Vectors. If there's a typo or a correction needs to be made, feel free to email me at benjamin.smidt@utexas.edu so I can fix it. Thank you! I hope you find this document helpful :).

Contents

1 Neural Machine Translation with RNNs

1.1 Problem G

Question:
The *generate-sent-masks()* function in *nmt-model.py* produces a tensor called *enc-masks*. It has shape (batch size, max source sentence length) and contains 1s in positions corresponding to 'pad' tokens in the input, and 0s for non-pad tokens. Look at how the masks are used during the attention computation in the step() function. First explain (in around three sentences) what effect the masks have on the entire attention computation. Then explain (in one or two sentences) why it is necessary to use the masks in this way.

Answer:
You can see in the code that we set the attention weights for the masked values (wherever 'pad' token appears) to negative infinity. As a result they are given zero weight in the softmax function, meaning the network never attends to these words. This is what we want because we don't want the model to use the 'pad' tokens for prediction since they're just a filler value.

We have to use the masks in this manner because we need to keep our vector shapes but we don't want to attend the 'pad' tokens. By setting their weights to negative infinity before the softmax, we effectively cut them out of the attention mechanism while retaining the vector shapes we need for the network to function properly.

1.2 Problem I