NLP Homework 5

Due 7/20

Watch Video: <https://www.youtube.com/watch?v=rBCqOTEfxvg>

Slides: <https://drive.google.com/file/d/0B8BcJC1Y8XqobGNBYVpteDdFOWc/view>

**Q1: Why is the attention faster than a recurrent when the attention scales as n2d while Recurrent is nd2? (n is sequence length, d is the dimension)?**

In the RNN model, we have an input vector which is appended to the output of the previous layer (num\_neurons). This sum is the ‘d’ for RNN. This is then multiplied by the Weight Matrix which must also have one dimension as ‘d’ in order to carry out the multiplication. Hence, we have a operation at each step of the sequence. Then if we have ‘n’ steps in the sequence, each step must wait for the previous step to execute before that step can be performed. Hence, this is sequential and the final complexity scales as .

For the attention model, this is different. Here we have an attention at each time step and each of these attentions gets its inputs from each of the time steps. Hence, if we have a sequence length of ‘n’, this is a operation. Then each of these operations is just a dot product of the dimension ‘d’ in the model. Hence the complexity is Also, all these are independent matrix multiplications and can be parallelized, hence we don’t add any further term to the complexity.

Now, the value of ‘d’ is generally much greater than ‘n’. So RNN that have a term in the complexity is more costly compared to Attention which has a term in the complexity.

**Q2: What is label smoothing?**

Label smoothing is a regularization technique used in classification problems. It also helps to prevent an overconfident model where the prediction probability is much higher than what is truly observed (i.e. model is not calibrated).

Without label smoothing, if the classifier predicts the right class, then we don’t add any cost to the overall loss of the network. If it predicts the wrong class, then we add maximum possible cost to the overall loss function. In a way, this incentivizes the neural network to memorize the data even if that means that it does not generalize well to unseen data in the future.

Label smoothing works by adding a small cost to the overall loss even if the network makes a correct prediction. On the other extreme, when the network makes a wrong prediction, it does not add the full cost to the overall loss (instead it adds a little less). This is turn gives the neural network some leeway in making mistakes during training if that results in better generalization.

Label smoothing is obtained by replacing the one hot encoded (OHE) labels with a smoothed version given by:

where is the smoothing factor and K is the number of labels.

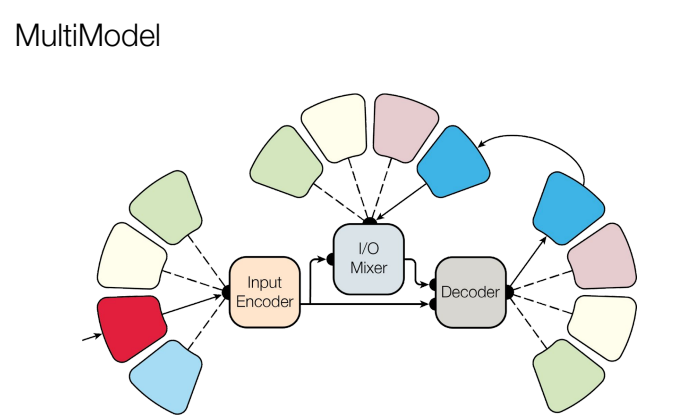
When , this is the same as having OHE as labels. When , this is equivalent to having all outputs having a label = .

In reality, is a small value so for labels that were encoded OHE 1, the smoothed label will be slightly less than 1 and for labels that were OHE 0, the smoothed label will be slightly greater than 0. Hence, when the network correctly predicts a OHE 1 label, because the smoothed label is slightly less than 1, we still add a small cost to the overall loss (similarly for OHE 0 label). On the other hand, when the network incorrectly predicts an OHE 1 label as OHE 0, we don’t add the full cost since the smoothed label is not 1 but slightly less than 1.

References: [Link1](https://towardsdatascience.com/what-is-label-smoothing-108debd7ef06) | [Link2](https://towardsdatascience.com/label-smoothing-making-model-robust-to-incorrect-labels-2fae037ffbd0)

**Q3: How was the Transformer extended to other tasks beside translation?**

The transformer was extended to several other tasks (besides translation) such as image classification, image captioning, parsing sentences to assign Parts of Speech (POS) and speech recognition. However, all these tasks had very different inputs and outputs. Example, the image might be 256x256 while the text data might be a group of word vectors. Since the model had to be trained with all these datasets simultaneously, the authors had to make sure that they can standardize any inputs going into the network and any outputs coming out of the network. In order to do so, the authors built several modalities that processed the raw inputs and outputs and converted them into a common format that could be digested by the network (image shown below from original talk).



In addition, the model did not do so well out of the box, so the authors had to increase the size of the network. But in order to not cause the network to slow down, they used a Mixture of Models (MOE) approach where there is a gate and that gate only pick a subset of matrices for each operation and a different subset for the next operation. This MOE model will learn which subset of the matrices to pick for each operation. Since the MOE model can train several matrices to learn many features, but at the same time it only operates on a subset of these matrices, it is able to achieve better results without sacrificing speed.

**Q4: What 8 tasks was the multimodal trained on?**

The multimodal model was trained on the following 8 tasks

1. 4 translation tasks( English to German and vice versa, English to French and vice versa),
2. Image classification (ImageNet),
3. Image Captioning (COCO),
4. Parsing (Part of Speech Tagging) from WSJ, and
5. Speech recognition (PTB)

**Q5: What package does the author talk about that contains the Transformer Code?**

The package that contains the Transformer Code is *tensor2tensor* and can be found [here](https://github.com/tensorflow/tensor2tensor). It has all the tricks that the authors applied when training their networks (learning rate decay, label smoothing, Adam optimizer settings, etc.) and hence is expected to run smoother out of the box compared to someone trying to implement these from scratch.