Q1: Encoder-Decoder models

We've seen how encoder-decoder architectures can be used for natural language translation. Give another example of a task that encoder-decoder models would be useful for.

Q2: Attention practice

Given a set of key vectors $\{k_1 = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}, k_2 = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}, k_3 = \begin{bmatrix} -2 \\ -3 \\ 0 \end{bmatrix}\}$ and query vector $q = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$, compute the attention weighted sum of keys using dot-product similarity $sim(x, y) = x^T y$

Q3: Similarity scores

A scoring function is used to measure similarity between two vectors, which then determines the attention weight between query and key. Compare and contrast each of the different scoring functions given in the lecture slides. How do you think these differences would affect an attention model?

Q4: Distance scores

Usually attention weights are computed based on similarity between query and key. Would it be possible to instead compute attention weights based on the Euclidean distance between query and key, so that vectors which are further apart have smaller weight? If so, then explain how you would do it. If not, then explain what trouble you would run into.

Q5: SoftMax normalisation

a) Attention weights are run through a SoftMax function to ensure they sum up to 1. Explain why this step is important.
b) Would the attention mechanism still work if we did the normalisation as follows?

$$\boldsymbol{a}_i = \frac{\boldsymbol{relu}(\tilde{\boldsymbol{a}}_i)}{\sum_{j=1}^{n} \boldsymbol{relu}(\tilde{\boldsymbol{a}}_j)}$$

Q6: Tree based Encoder-Decoder (Challenge Question)

Suppose that we have a dataset where both the inputs and outputs are binary trees. Each node in the tree contains a vector of features which represent that tree. The predicted trees should have the same structure as the target trees, and all of the predicted nodes should have the same vectors as the target nodes.

a) Describe how you would design an encoder-decoder model for this dataset.
b) How would your model need to change if the trees were not binary, so that each node could have any number of children?

Practical Exercise: Memory Test

The Memory_Test.ipynb notebook contains code for running a very simple “memory test” on various recurrent neural network models. The models are run on a synthetic dataset, where inputs are sequences of random vectors and the label of a given sequence is the first vector of that sequence. In order to solve this task the model must remember the first vector it sees and then ignore everything else. The notebook runs this test with sequences of varying sizes and records how the models perform.

Using the provided code try to answer the following questions:

a) Which model has the longest memory?
b) How does hidden dimension of the model affect memory?
c) How does learning rate affect memory?
d) Implement the proposed ReLU normalisation from Q5b) in the AttentionGRU from models.py. Does the new attention perform as you predicted in Q5b)?