AI Benchmarks and Exercises

- The Copying Task

The Copying Task

The Copying Task is a benchmarking task in NLP that assesses recurrent models (and other sequential models)' ability to retain information for long sequence lengths. Before transformers, RNNs like LSTM or GRU suffered from the vanishing gradient problem which means that they become less effective as the length of the sequence increases.
The copying task can be used to illustrate a model's ability to handle long sequences and see how well it deals with the increasingly long sequences of letters/characters/whatever.
In principle, we show the model a sequence of letters it has to remember, then a long blank sequence and then ask it to spit out the original sequence of characters that we actually care about at the end.

Task Definition

In the paper they use ***a*** to define the alphabet of letters that we care about:

- The first 8 items are actual letters that we might care about,
- The 9th item is considered 'blank' and just used to pad out the sequence and make it long for tests
- The 10th item is a 'delimiter' character to tell the model to start spitting out the original sequence.

$\{a_i\}_{i=0}^{9}$ the full vector a and all possible values

$\{a_i\}_{i=0}^{7}$ the 'alphabet' of letters we care about

a_8 the 'blank' padding character

a_9 the delimiter

We can define different lengths of sequence that we want to test against - in the paper they use ***T*** to represent this value. For example, T=100 means "***define a sequence, run 100 blanks through the system, then have it recall the sequence from the beginning***".
To set up the test we:

1. use the characters from ***a*** 0 to 7 to generate a random sequence of characters that needs to be memorised - this sequence is 10 characters long. We sample **with** replacement which means letters are allowed to repeat.
2. Generate T ***blank*** characters
3. Use the delimiter to signal to the model that it should generate the remaining steps
4. Have the model infer the sequence - since the length of the sequence is known, we can halt after the correct number of characters have been outputted and check to make sure that it is correct.

Our sequence would look something like this.



Scaling Characteristics

- We can trivially generate 1000s of sequences like this and use it as a simple benchmark. We want to train on a diverse set of sequence examples so that the model doesn't over-fit and learn to always predict the same sequence.
- As we are sampling with replacement, there are 8^10, or approximately ***1.1 Billion*** possible problems and solutions when the length of the input sequence is 10. As long as we are using relatively small models, it shouldn't be possible for the model to memorise all possible solutions in its weights. Modern transformers have multiple billion parameters and therefore could feasibly overfit to this problem if trained on all possible solutions.

References

Arjovsky, Martin, Amar Shah, and Yoshua Bengio. 'Unitary Evolution Recurrent Neural Networks'