TF2RNN

TL;DR

An attempt to distill transformer-based language models into recurrent neural nets.

Why?

Transformer-based architectures are distinct from recurrent neural networks (RNNs) in that they look at the entire sequence in one step. This makes for great encoders. A sequence goes in, an embedding comes out. RNNs can do the same, in principle, but they look at one token at a time. An RNN sequence encoder generates the embedding over multiple steps.

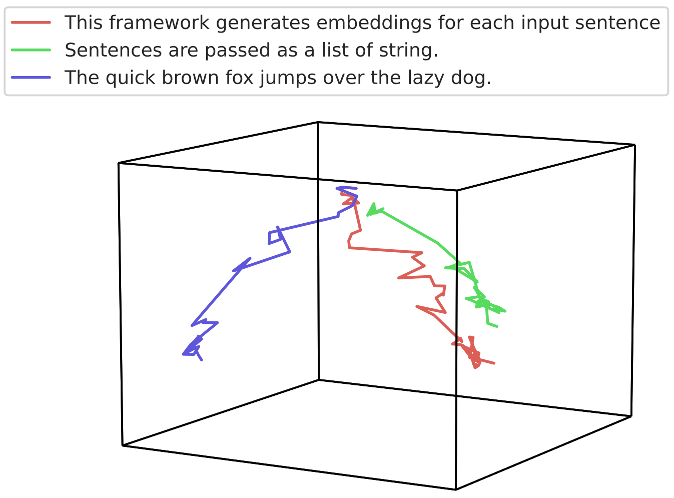
The distinction between transformers and RNNs is sometimes an advantage and sometimes a disadvantage. During training, it is often easier to train a transformer. An RNN's feedback signal (i.e., learning signal) has to travel back through every step taken by the RNN. Two consequences of this are: vanishing or exploding gradients, and chaotic loss-landscapes. During inference, RNNs have the advantage. Since they process things sequentially, they do not have to re-compute the entire embedding every time a new token is made available. Furthermore, unlike standard self-attention, an RNN's computational cost grows linearly w.r.t. sequence length.

Transformers are easier to train, but RNNs are cheaper to deploy. So, what if we could distill a transformer into an RNN?

How?

Consider the sentence "Hello world". If the sentence is made available to a transformer sequentially, it produces an embedding for each incomplete stage. For instance, "Hello wor" results in an intermediate embedding, "Hello worl" in another, and "Hello world" in the final embedding. Text Embeddings Reveal (Almost) As Much As Text shows that these embeddings are sufficient to reconstruct the original sequence, up to a certain length. After a certain length, some details begin to get lost. These embeddings are doing what an RNN's recurrent state is supposed to do. Hypothetically, one could train RNNs to work as transition functions between intermediate embeddings. If we already have the right embeddings, then there is no need to back-propagate through time. So, training the RNN should not be any more difficult than training a typical multilayer perceptron (MLP).

Example



This plot was generated with hypertools. We took a sentence transformer and used it to produce trajectories in embedding space by unmasking sentences incrementally, one character at a time. The question is whether it is feasible to train an RNN to follow trajectories like these and generalize from them.

(Preliminary) Results

Coming soon