LSTM Decoder

In the project, we pass all our inputs as a sequence to an LSTM. A sequence looks like this: first a feature vector that is extracted from an input image, then a start word, then the next word, the next word, and so on!

Embedding Dimension

The LSTM is defined such that, as it sequentially looks at inputs, it expects that each individual input in a sequence is of a **consistent size** and so we *embed* the feature vector and each word so that they are `embed_size`.

Sequential Inputs

So, an LSTM looks at inputs sequentially. In PyTorch, there are two ways to do this.

The first is pretty intuitive: for all the inputs in a sequence, which in this case would be a feature from an image, a start word, the next word, the next word, and so on (until the end of a sequence/batch), you loop through each input like so:

```
for i in inputs:
    # Step through the sequence one element at a time.
    # after each step, hidden contains the hidden state.
    out, hidden = lstm(i.view(1, 1, -1), hidden)
```

The second approach, which this project uses, is to **give the LSTM our *entire sequence*** and have it produce a set of outputs and the last hidden state:

```
# the first value returned by LSTM is all of the hidden states throughout
# the sequence. the second is just the most recent hidden state
```

```
# Add the extra 2nd dimension
inputs = torch.cat(inputs).view(len(inputs), 1, -1)
hidden = (torch.randn(1, 1, 3), torch.randn(1, 1, 3))  # clean
out hidden state

out, hidden = lstm(inputs, hidden)
```

LSTM/RNN revision and building from scratch:

- https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html
- http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/*** amazing post. Need to implement each bit of it.
- Colah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/