Long sequences in Tensorflow/Keras

Dealing with something as "simple" as sequences can be surprisingly difficult in Tensorflow/Keras.

- One is required to manually break up long sequences into multiple, shorter subsequences
- The ordering of the examples in a mini-batch now becomes relevant

Consider a long sequence $\mathbf{x}^{(\mathbf{i})}$ of length T.

The "natural" way to represent this $\mathbf{X}$ is

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)}_{(1)} & \mathbf{x}^{(1)}_{(2)} & \dots & \mathbf{x}^{(1)}_{(T^{(1)})} \\ \mathbf{x}^{(2)}_{(1)} & \mathbf{x}^{(2)}_{(2)} & \dots & \mathbf{x}^{(2)}_{(T^{(2)})} \\ \vdots & & & \end{pmatrix}$$

for equal example sequence lengths $T = T^{(1)} = T^{(2)} \dots$

Suppose that the example sequence lengths T is too long (e.g., exhausts resources)

In that case, each example needs to be broken into *multiple* "child-examples" of shorter length T'.

There will be T/T' such child examples, each having a subsequence of length T' of the parent example's sequence.

We write $\mathbf{x}^{(i,\alpha)}$ to denote child example number α of parent examples i.

- The elements of $\mathbf{x}^{(i,\alpha)}_{(t)}$ are $[\ \mathbf{x}^{\mathbf{(i)}}_{(t)} \mid ((\alpha - 1) * T/T') + 1 \leq t \leq (\alpha * T/T')\ ]$.
- The subsequence $\mathbf{x}^{(i,\alpha+1)}_{(t)}$ starts right after the end of subsequence $\mathbf{x}^{(i,\alpha)}_{(t)}$

Great care must be taken when arranging child examples into a new training set $\mathbf{X}'$.

This is because of the relationship between examples that TensorFlow implements (as of the time of this writing)

- Examples *within* a minibatch are considered independent
    - May be evaluated in parallel
    - So *not* suitable to place two children of the same parent in the same minibatch
- Example i of consecutive mini-batches *can* be made dependent
    - With an optional flag

To get adjacent subsequences of one sequence to be treated in the proper order by TensorFlow:

- Define the number of minibatches to be T/T', which is the number of subsequences
- Each subsequence of example i should be at the *same position* within each of the n/n' minibatches
- Set RNN optional parameter `stateful=True`
- When fitting the model: set `shuffle=False`

$$\text{Minibatch } 1 = \begin{pmatrix} \mathbf{x}^{(1)}_{(1)} & \mathbf{x}^{(1)}_{(2)} & \dots & \mathbf{x}^{(1)}_{(T')} \\ \mathbf{x}^{(2)}_{(1)} & \mathbf{x}^{(2)}_{(2)} & \dots & \mathbf{x}^{(2)}_{(T')} \\ \vdots & & & \end{pmatrix}$$

$$\text{Minibatch 2} = \begin{pmatrix} \mathbf{x}^{(1)}_{(T'+1)} & \mathbf{x}^{(1)}_{(T'+2)} & \cdots & \mathbf{x}^{(1)}_{(T'+T')} \\ \mathbf{x}^{(2)}_{(T'+1)} & \mathbf{x}^{(2)}_{(T'+2)} & \cdots & \mathbf{x}^{(2)}_{(T'+T')} \\ \vdots & & & \end{pmatrix}$$

Thus, row i of minibatch b corresponds to child α of parent example i

Why does this work ?

The flag `stateful=True`

- Tells TensorFlow to **not** reset the latent state of the RNN at the start of a new minibatch
    - When examples across batches are *independent*, the RNN should begin from step 1
    - And therefore re-initialize the latent state

By arranging the minibatches as we have

- The latent state of the RNN when processing child $(\alpha + 1)$ of example i
- Is the latent state of the RNN after having process the subsequence of child α of example i

The flag `shuffle=False`

- Tells TensorFlow to **not** shuffle the examples in the minibatches
- In order to preserve the fact that row i of each minibatch is a different child of the same parent

Conclusion

Long sequences present some technical issues in Keras and other frameworks.

We recognize that mitigating the issues was a highly technical topic that might take some effort to absorb.

We hope that, eventually, a better API might alleviate the burden for the end user.

In [3]:

```
print("Done")
```

```
Done
```