Predictions

Create word predictions based on the output of your LSTM model.

Chapter Goals:

- Calculate softmax probabilities to make predictions for each time step

A. Calculating probabilities

When using a feed-forward neural network for multiclass predictions, we first need to convert the model's logits into probabilities with the softmax function. For an RNN language model, the idea is the exact same. After converting the RNN's outputs into logits, we apply the softmax function to the *final dimension* of the logits.

The model we've built so far in this section predicts the likely next word given an input sentence (see the **Language Model** chapter for a refresher)

```
import tensorflow as tf
# Logits with a vocab_size = 100
logits = tf.placeholder(tf.float32, shape=(None, 5, 100))
probabilities = tf.nn.softmax(logits, axis=-1)
```

Using the softmax function to calculate probabilities from logits.

By applying the softmax function to the final dimension of the logits, we calculate probabilities for every vocabulary word at each time step in each sequence. In the example above, the first dimension of `probabilities` corresponds to each sequence in the batch, the second dimension corresponds to the time steps, and the third dimension corresponds to the vocabulary word probabilities.

B. Word predictions

Similar to regular multiclass predictions with an MLP (Multilayer Perceptron)

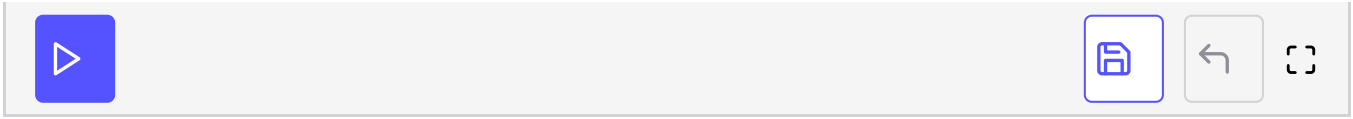
Similar to regular multiclass predictions with an MLP (Multilayer Perceptron), we calculate the RNN's word predictions by taking the highest probability

word at each time step. The nice thing about how we tokenized our text sequences is that each vocabulary word becomes a positive integer. This means that, for each time step, the index containing the highest probability will be the model's word prediction.

The example below shows how to calculate word predictions from probabilities. We use `tf.argmax` to retrieve the indexes of the final dimension that contain the highest probabilities.

```
import tensorflow as tf
# Placeholder for the model probabilities
probabilities = tf.placeholder(tf.float32, shape=(None, 5, 100))

word_preds = tf.argmax(probabilities, axis=-1)
```



C. Current state of the art

LSTMs are more effective at natural language processing than conventional neural networks, but more effective models do exist.

Currently, the best performing model uses transformers, which are detailed in this paper.

To play around with a model that uses transformers, click this link.