Logits

Calculate logits based on the final output of the model.

Chapter Goals:

- Calculate logits from the combined BiLSTM outputs

A. Concatenation

As mentioned in the previous chapter, the BiLSTM returns two outputs: the forwards and backwards outputs. In order to calculate the model's logits, we need to combine these two outputs. We do this through simple *concatenation*.

Concatenation in TensorFlow refers to appending tensors along a certain dimension. The function that performs this operation is `tf.concat`. It takes in two required arguments: a list of tensors to concatenate and the axis (dimension) to concatenate along.

Below we demonstrate an example usage of `tf.concat`. The variables `o1` and `o2` are NumPy arrays representing the concatenation outputs.

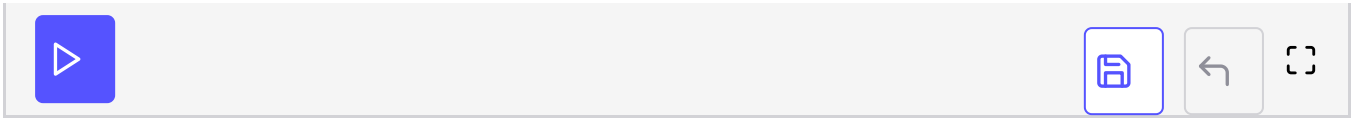
```
import tensorflow as tf
# Shape: (2, 2, 3)
t1 = tf.constant([
    [[1, 2, 3], [4, 5, 6]],
    [[0, 4, 8], [3, 2, 2]]
])

# Shape: (1, 2, 3)
t2 = tf.constant([
    [[9, 9, 9], [8, 8, 8]]
])

# Shape: (2, 2, 2)
t3 = tf.constant([
    [[9, 9], [1, 1]],
    [[7, 2], [8, 8]]
])

with tf.Session() as sess:
    o1 = sess.run(tf.concat([t1, t2], 0))
    o2 = sess.run(tf.concat([t1, t3], -1))
```

```
print(repr(o1))
print(repr(o2))
```



When concatenating tensors, each tensor needs to have the exact same shape, apart from the axis that's being concatenated. The tensors are concatenated in the same order that they appear in the list.

We can use `-1` for the second argument to specify the final tensor dimension as the axis of concatenation. This is a useful shortcut for concatenating along the final dimension, and it is how we combine the BiLSTM outputs.

B. Final time step

Unlike the language model from the **Language Model** section of this course, when we create an LSTM for classification we only use the final time step output for each sequence in the batch. This is because we take into account the entire text sequence for classification, whereas for language modeling we were focused on completing partial sequences.

So after combining the forwards and backwards LSTM outputs, we retrieve the final time step values (using `tf.gather_nd`), and then pass those values through a final fully-connected layer to obtain the model's logits.

Time to Code!

In this chapter you'll be completing the `calculate_logits` function, which calculates logits based on the outputs of the BiLSTM.

The function's input, `lstm_outputs`, is a tuple containing the outputs of the forwards and backwards LSTMs. Our first step is to separate the tuple into two distinct variables.

Set `lstm_outputs_fw, lstm_outputs_bw` equal to `lstm_outputs`.

The way we combine the two LSTM outputs is by concatenating the output values along their final dimension. The input list for `tf.concat` should be `[lstm_outputs_fw, lstm_outputs_bw]`.

Set `combined_outputs` equal to `tf.concat` applied with the specified input list as the first argument and `-1` as the second argument.

We provide a function, `get_gather_indices`, which uses code from the **Language Model** section to calculate the indices of each sequence's final time step. Use that function, along with `tf.gather_nd`, to retrieve the final time step values from `combined_outputs`.

Set `gather_indices` equal to `self.get_gather_indices` applied with `batch_size` and `sequence_lengths` as arguments.

Set `final_outputs` equal to `tf.gather_nd` applied with `combined_outputs` and `gather_indices` as arguments.

Since our task is binary text classification, we use a final fully-connected layer with a single node to obtain the model's logits.

Set `logits` equal to `tf.layers.dense` applied with `final_outputs` as the first argument and `1` as the second argument. Then return `logits`.

```
import tensorflow as tf
tf_fc = tf.contrib.feature_column

# Text classification model
class ClassificationModel(object):
    # Model initialization
    def __init__(self, vocab_size, max_length, num_lstm_units):
        self.vocab_size = vocab_size
        self.max_length = max_length
        self.num_lstm_units = num_lstm_units
        self.tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=self.vocab_size)

    def get_gather_indices(self, batch_size, sequence_lengths):
        row_indices = tf.range(batch_size)
        final_indexes = tf.cast(sequence_lengths - 1, tf.int32)
        return tf.transpose([row_indices, final_indexes])

    # Calculate logits based on the outputs of the BiLSTM
    def calculate_logits(self, lstm_outputs, batch_size, sequence_lengths):
        # CODE HERE
        pass
```