Loss

Calculate the model's sigmoid cross entropy loss.

Chapter Goals:

- Calculate the model's loss using sigmoid cross entropy

A. Sigmoid cross entropy

The task for our model is to classify input text sequences as either negative (label `0`) or positive (label `1`). This is equivalent to binary classification. As with regular binary classification, we use sigmoid cross entropy to calculate the model's loss.

For an in-depth and intuitive explanation of sigmoid cross entropy and binary classification, check out the Machine Learning for Software Engineers course on Educative.

Time to Code!

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

The first step to calculating the model's loss is to first calculate the logits. We can use the `calculate_logits` function we completed in the previous chapter.

Set `logits` equal to `self.calculate_logits` applied with `lstm_outputs`, `batch_size`, and `sequence_lengths` as arguments.

Since we're performing binary classification, we use sigmoid cross entropy for the loss. We also need to convert the integer labels into floats.

Set `float_labels` equal to `tf.cast` applied with `labels` as the first argument and `tf.float32` as the second argument.

Set `batch_loss` equal to `tf.nn.sigmoid_cross_entropy_with_logits` applied with `float_labels` and `logits` for the `labels` and `logits` keyword

arguments, respectively.

The output of the function is the overall aggregate loss. This means we need to sum together each individual sequence's loss in the batch.

Set `overall_loss` equal to `tf.reduce_sum` applied to `batch_loss`. Then return `overall_loss`.

```
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):
        lstm_outputs_fw, lstm_outputs_bw = lstm_outputs
        combined_outputs = tf.concat([lstm_outputs_fw, lstm_outputs_bw], -1)
        gather_indices = self.get_gather_indices(batch_size, sequence_lengths)
        final_outputs = tf.gather_nd(combined_outputs, gather_indices)
        logits = tf.layers.dense(final_outputs, 1)
        return logits

    # Calculate the loss for the BiLSTM
    def calculate_loss(self, lstm_outputs, batch_size, sequence_lengths, labels):
        # CODE HERE
        pass
```