```
# import libraries
try:
  # %tensorflow_version only exists in Colab.
  !pip install tf-nightly
except Exception:
  pass
import tensorflow as tf
import pandas as pd
from tensorflow import keras
!pip install tensorflow-datasets
import tensorflow_datasets as tfds
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)


# get data files
!wget https://cdn.freecodecamp.org/project-data/sms/train-data.tsv
!wget https://cdn.freecodecamp.org/project-data/sms/valid-data.tsv

train_file_path = "train-data.tsv"
test_file_path = "valid-data.tsv"


train = pd.read_csv(train_file_path,sep='\t',header=None)
test = pd.read_csv(test_file_path,sep='\t',header=None)

print(train[0][0]) # column 0 = ham/spam, colum 1 = text
train[0] = pd.factorize(train[0])[0]
test[0] = pd.factorize(test[0])[0]
print(train[0][0]) # ham = 0, spam = 1


train_tensordata = tf.data.Dataset.from_tensor_slices((train[1].values, train[0].values)) # m
test_tensordata = tf.data.Dataset.from_tensor_slices((test[1].values, test[0].values))

test_tensordata.element_spec




BUFFER_SIZE = 10000
BATCH_SIZE = 64


train_tensordata = train_tensordata.shuffle(BUFFER_SIZE).batch(BATCH_SIZE).prefetch(tf.data.A
test_tensordata = test_tensordata.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
print(train_tensordata)
```

```
VOCAB_SIZE = 1000
encoder = tf.keras.layers.TextVectorization(
    max_tokens=VOCAB_SIZE)
encoder.adapt(train_tensordata.map(lambda text, label: text))


vocab = np.array(encoder.get_vocabulary())
vocab[:20]


model = tf.keras.Sequential([
    encoder,
    tf.keras.layers.Embedding(
        input_dim=len(encoder.get_vocabulary()),
        output_dim=64,
        # Use masking to handle the variable sequence lengths
        mask_zero=True),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])


print([layer.supports_masking for layer in model.layers])

# predict on a sample text without padding.
sample_text = ('The movie was cool. The animation and the graphics '
               'were out of this world. I would recommend this movie.')
predictions = model.predict(np.array([sample_text]))
print(predictions[0])

# predict on a sample text with padding

padding = "the " * 2000
predictions = model.predict(np.array([sample_text, padding]))
print(predictions[0])


model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])


history = model.fit(train_tensordata, epochs=10,
                    validation_data=test_tensordata,
                    validation_steps=30)


test_loss, test_acc = model.evaluate(test_tensordata)

print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)
```

```
pred_text = "how are you doing today?"
predictions = model.predict(np.array([pred_text]))
predictions


# function to predict messages based on model
# (should return list containing prediction and label, ex. [0.008318834938108921, 'ham'])
def predict_message(pred_text):
  predictions = model.predict(np.array([pred_text]))
  if predictions[0][0] < 0.5:
    prediction = [predictions[0][0], 'ham']
  else:
    prediction = [predictions[0][0], 'spam']

  return (prediction)

pred_text = "how are you doing today?"

prediction = predict_message(pred_text)
print(prediction)


# Run this cell to test your function and model. Do not modify contents.
def test_predictions():
  test_messages = ["how are you doing today",
                   "sale today! to stop texts call 98912460324",
                   "i dont want to go. can we try it a different day? available sat",
                   "our new mobile video service is live. just install on your phone to start
                   "you have won £1000 cash! call to claim your prize.",
                   "i'll bring it tomorrow. don't forget the milk.",
                   "wow, is your arm alright. that happened to me one time too"
                  ]

  test_answers = ["ham", "spam", "ham", "spam", "spam", "ham", "ham"]
  passed = True

  for msg, ans in zip(test_messages, test_answers):
    prediction = predict_message(msg)
    if prediction[1] != ans:
      passed = False

  if passed:
    print("You passed the challenge. Great job!")
  else:
    print("You haven't passed yet. Keep trying.")

test_predictions()
```