```
import os, re, sys
import tensorflow as tf
import numpy as np
import rnn
import preprocess
import random

# Yields (batch_size) examples, truncated at (max_time) words.
# Examples are randomly pulled from (input_directory) and
# Translated to integers using (emb_dict)
def dataset(input_directory, batch_size, emb_dict, max_time):
  file_gen = os.listdir(input_directory)
  random.shuffle(file_gen)
  batch_num = 0
  inputs = np.zeros((batch_size, max_time))
  targets = np.zeros((batch_size,2))
  sequence_length = np.zeros((batch_size))
  for name in file_gen:
    file_path = input_directory + '/' + name
    f = open(file_path)
    w = preprocess.tokenize(f.read())
    rating = int(re.sub('_|\.txt',' ',name).split()[1])
    targets[batch_num][0:2] = [0,1] if rating < 5 else [1,0]
    sequence_length[batch_num] = len(w)
    for time_num in range(min(max_time,len(w))):
      inputs[batch_num][time_num] = emb_dict.get(w[time_num],0)

    batch_num += 1
    if batch_num == batch_size:
      yield inputs,targets,sequence_length
      batch_num = 0
      inputs = np.zeros((batch_size, max_time))
      targets = np.zeros((batch_size,2))
      sequence_length = np.zeros((batch_size))

batch_size = 1000  # Number of reviews to consider at once
max_time = 1024   # Maximum number of words in a given review
# Load the word embedding
emb_dict = np.load('emb_dict.npy').item()
embeddings = np.load('final_embeddings.npy')

for hidden_size,lr in zip([2,4,8,16,32,64,128]*2,[0.01,0.001]*7):
  # Reset the TensorFlow graph
  g = tf.Graph()
  tf.reset_default_graph()
  with g.as_default():
    # Set global step to zero, for keeping track of training progress
    global_step_tensor = tf.Variable(0, trainable = False, name = 'glo
bal_step')
    # Make the RNN
    r = rnn.classifier(
      batch_size = batch_size,
      learning_rate = lr,
      hidden_size = hidden_size,
      max_time = max_time,
      embeddings = embeddings,
      global_step = global_step_tensor
```

```
    )

    # Training session
    with tf.Session() as sess:
      saver = tf.train.Saver(max_to_keep = 200)
      train_writer = tf.summary.FileWriter(sys.argv[1]+'/train_'+str(h
idden_size)+'_'+str(int(lr*1000)))
      test_writer = tf.summary.FileWriter(sys.argv[1]+'/test_'+str(hid
den_size)+'_'+str(int(lr*1000)))
      sess.run(tf.global_variables_initializer())
      for epoch in range(50):
        saver.save(sess, './'+sys.argv[2]+'_'+str(hidden_size)+'_'+str
(int(lr*1000))+"/imdb-rnn-e%d.ckpt"%epoch)
        print('epoch: ',epoch)
        # Testing #
        sess.run(tf.local_variables_initializer())
        if not (epoch % 5):
          for inputs,targets,sequence_length in dataset(\
              './aclImdb/test/posneg/',batch_size,emb_dict,max_time):
            accuracy,global_step,summary = sess.run([r.update_accuracy
, global_step_tensor, r.merged],
              feed_dict={r.inputs:inputs,r.targets:targets,r.sequence_
length:sequence_length,
                         r.keep_prob:1.0})
          test_writer.add_summary(summary, global_step)
        # Training #
        sess.run(tf.local_variables_initializer())
        for inputs,targets,sequence_length in dataset(\
            './aclImdb/train/posneg/',batch_size,emb_dict,max_time):
          loss,updates,embed,output,updated_accuracy,\
          mean,logits,prob,sequence_length,\
          probe,labels,global_step,learning_rate,summary = \
            sess.run([r.loss,r.updates,r.embed,r.output,r.update_accur
acy,\
                  r.mean,r.logits,r.probability,r.sequence_length,\
                  r.probe,r.targets,global_step_tensor,r.learning_rate
,r.merged],
              feed_dict={r.inputs:inputs,r.targets:targets,r.sequence_le
ngth:sequence_length,
                         r.keep_prob:0.5})
        train_writer.add_summary(summary, global_step)
```