โครงงานที่ 4: การสร้าง Sentiment Classifier ด้วยชุดข้อมูลรีวิวภาพยนตร์

รหัสนักศึกษา _____57130500098_____ ชื่อ-นามสกุล _____กรรชัย สดหอม_____

เป้าหมายของโครงงาน:

สร้างโปรแกรม recurrent neural network ที่สามารถจำแนก ข้อความรีวิวภาพยนตร์ ออกเป็น POSITIVE และ NEGATIVE

ชุดข้อมูล:

http://ai.stanford.edu/~amaas/data/sentiment/acllmdb_v1.tar.gz

ตัวอย่างโปรแกรม:

- DL4J: https://goo.gl/EjeP4k
- Keras: https://github.com/keras-team/keras/blob/master/examples/imdb_lstm.py

1. **การทำความเข้าใจข้อมูล (Data Understanding)**

จะเป็นข้อมูลการรีวิวของหนังว่าแต่ละ Comment ที่ Review ไปนั้นมีค่าความเป็น Positive หรือ Negative เพื่อใช้ดูแนวโน้มของหนังว่ามีแนวโน้มที่จะดีหรือไม่ดีได้ โดยข้อมูลที่เก็บนั้นก็จะแบ่งเป็น Folder โดยแยกเป็น Train Data กับ Test Data และในแต่ละ Folder แยกก็จะมี Folder ข้างในนั้นอีกเป็น Positive กับ Negative ซึ่งจะเป็นข้อความไว้ ซึ่ง Label ก็จะมี Positive กับ Negative

2. **การเตรียมข้อมูล (Data Pre-processing)**

ข้อมูลที่ใช้สำหรับการเรียนรู้ก็จะมี 25,000 Sequences และข้อมูลที่ใช้สำหรับการทดสอบนั้นก็จะมี 25,000 Sequences เช่นกัน และข้อมูลที่ได้มาจาก Keras นั้นก็จะมีการแปลงเป็นรูปแบบของ Vector

3. **การสร้างโมเดลเครือข่ายประสาทเทียม (Model Building)**

```
print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.25, recurrent_dropout=0.5))
model.add(Dense(1, activation='tanh'))
```

```
Build model...
```

- จาก Source Code ก็จะบอกว่า Embedding โดยที่ max_features นั้นมีค่า 20,000 และ batch_size เท่ากับ 32 คือเพิ่มขนาดขึ้นมา
- มี LSTM Layer ขึ้นมาซึ่งนำข้อมูลที่ผ่านกระบวนมาแล้วมาต่อเป็นรูปแบบ Sequences ซึ่งกำหนดขนาดที่ 128 แล้ว dropout กำหนดค่า prop เป็น 0.25 แล้วก็ recurrent_dropout เป็น 0.5
- สุดท้าย Dense Layer นั้นมีขนาด Output เป็น 1 ก็สามารถบอกได้ว่าข้อความนั้นเป็น Positive หรือ Negative โดยใช้ Activation Function เป็น Tanh

4. **การเทรนโมเดล (Model Training)**

4.1 Macbook Pro 2.7GHz - 128GB 2.7GHz dual-core Intel Core i5 processor (Turbo Boost up to 3.1GHz) with 3MB shared L3 cache and RAM 8 GiB with macOS High Sierra (version 10.13) and Jupyter Lab with Keras version 2.1.6

4.2 ค่า hyperparameter ต่างๆ

- Batch_size = 128
- Epochs = 5 รอบ

- Loss Function ใช้ binary_crossentropy
- Optimizer ใช้ adam

4.3 รายงานผลการเทรน โดยแสดงค่า Training Accuracy และระยะเวลาทั้งหมดที่ใช้ในการเทรนเน็ตเวิร์ก

```
print('Train...')
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=5,
          validation_data=(x_test, y_test))
score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

```
Train...
Train on 25000 samples, validate on 25000 samples
Epoch 1/5
25000/25000 [==============================] - 163s 7ms/step - loss: 0.1617 - acc: 0.8671 - val_loss: 0.75
27 - val_acc: 0.7605
Epoch 2/5
25000/25000 [==============================] - 166s 7ms/step - loss: 0.1623 - acc: 0.9210 - val_loss: 1.11
49 - val_acc: 0.7517
Epoch 3/5
25000/25000 [==============================] - 181s 7ms/step - loss: 0.1026 - acc: 0.9094 - val_loss: 1.21
56 - val_acc: 0.7464
Epoch 4/5
25000/25000 [==============================] - 204s 8ms/step - loss: 0.1122 - acc: 0.9348 - val_loss: 1.28
54 - val_acc: 0.7516
Epoch 5/5
25000/25000 [==============================] - 233s 9ms/step - loss: 0.0797 - acc: 0.9220 - val_loss: 1.32
28 - val_acc: 0.7452
```

ระยะเวลาที่ใช้ในการ Train Model นี้ก็คือ 984 วินาทีหรือประมาณ 16.4 นาที

5. การทดสอบประสิทธิภาพของโมเดล (Model Evaluation)

```
25000/25000 [==============================] - 37s 1ms/step
Test score: 1.322830154657364
Test accuracy: 0.74524
```

ประสิทธิภาพของโมเดลก็ Accuracy อยู่ที่ 0.7452 หรือ 74.52%

6. สรุปผลและข้อเสนอแนะ

- ใช้เวลาเทรนนานมากแค่ 5 Epoch เท่านั้น แล้วก็ค่า Loss ของตัว Model นั้นยังแอบมีปัญหาเรื่อง Overfitting อยู่ด้วย แนวทางการพัฒนาก็ได้ลองเทรนโมเดลแบบใช้ CNN ซึ่งได้ผลลัพธ์ที่ดีกว่าแล้วไวกว่าซึ่งอาจจะเป็นข้อเสนอแนะได้ ตามรูปด้านล่าง

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 400, 50)           250000
_________________________________________________________________
dropout_1 (Dropout)          (None, 400, 50)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 398, 250)          37750
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 250)               0
_________________________________________________________________
dense_1 (Dense)              (None, 250)               62750
_________________________________________________________________
dropout_2 (Dropout)          (None, 250)               0
_________________________________________________________________
activation_1 (Activation)    (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 251
_________________________________________________________________
activation_2 (Activation)    (None, 1)                 0
=================================================================
Total params: 350,751
Trainable params: 350,751
Non-trainable params: 0
_________________________________________________________________
```

```
# Train the model
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test),
          verbose=1)
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/4
25000/25000 [==============================] - 269s 11ms/step - loss: 0.4392 - acc: 0.7736 - val_loss: 0.2
920 - val_acc: 0.8785
Epoch 2/4
25000/25000 [==============================] - 255s 10ms/step - loss: 0.2424 - acc: 0.9014 - val_loss: 0.2
564 - val_acc: 0.8944
Epoch 3/4
25000/25000 [==============================] - 254s 10ms/step - loss: 0.1705 - acc: 0.9336 - val_loss: 0.2
814 - val_acc: 0.8871
Epoch 4/4
25000/25000 [==============================] - 253s 10ms/step - loss: 0.1175 - acc: 0.9580 - val_loss: 0.2
866 - val_acc: 0.8910
<keras.callbacks.History at 0x122f1c160>
```

```
score, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
preds = model.predict_classes(x_test, batch_size=batch_size)
```

```
25000/25000 [==============================] - 41s 2ms/step
```

```
print('Test accuracy:', acc)
print('Test score (loss):', score)
```

```
Test accuracy: 0.891
Test score (loss): 0.2865551609039307
```

****

การส่งงาน: โครงงานนี้เป็นโครงงานเดี่ยว นักศึกษาต้องส่ง

1. **รายงานในรูปแบบไฟล์ PDF (บันทึกชื่อเป็น imdb-report.pdf)**
2. **ซอร์สโค้ด**

โดยการอัพโหลดข้อมูลทั้งหมดไปไว้บน **Dropbox folder** ชื่อ **04_IMDB**

กำหนดส่ง: ภายใน 30 เมษายน 2561

****