21 - 1 딥러닝 및 응용 과제 2

텍스트 분류

조교 이봉석, 최원혁
tjr4090@hanyang.ac.kr
gandet09@hanyang.ac.kr

과제 개요

- **수업시간에 배운 다양한 방법들을 이용해서 텍스트 감성 분류 task 성능 향상.**
- **세부사항**
  - 제공된 base line 코드를 수정해서 **성능 향상**이 목표.
  - **긍정 부정 2종류**의 텍스트를 각각에 맞는 class로 **분류**하는 것이 목표.
  - 주어지는 데이터 (트레이닝 데이터 40,000개, 테스트 데이터 10,000개)
    - x_train.npy => 트레이닝 데이터, 전처리가 끝난 텍스트
    - y_train.npy => 트레이닝 데이터, 각 텍스트에 대한 라벨
    - x_test.npy => 성능 측정을 위한 test 데이터, 전처리가 끝난 텍스트
  - 트레이닝 데이터로 학습 후 x_test.npy을 분류한 **result를 이용해서 측정된 성능으로 점수 산출**.

제공된 코드

- **text_show.py : 주어진 전처리가 끝난 텍스트 데이터를 원래의 데이터로 보기 위한 코드**

```python
import numpy as np
import json

np_load_old = np.load

# modify the default parameters of np.load
np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)

x = np.load("data/x_train.npy")
y = np.load("data/y_train.npy")

np.load = np_load_old

index_to_word = json.load(open("data/dictionary.json"))
print(x[3])
# [1, 249, 1323, 7, 61, 113, 10, 10, 13, 1637, 14, 20, 56, 33, 24
print(' '.join([index_to_word[str(index)] for index in x[3]]))
# <sos> worst mistake of my life br br i picked this movie up at
print(y[3])
# 0
```

제공된 코드

- **eval.py : x_test 파일을 분류하여 나온 결과를 성능평가하기 위한 코드**

```
import sys
import numpy as np

result_path = sys.argv[1] # result.npy
GT = sys.argv[2] # data/y_test.npy

result = np.load(result_path)
y_prediction = np.argmax(result, 1)
y_test = np.load(GT)


is_correct = np.equal(y_prediction, y_test)
accuracy = np.mean(is_correct)

print(accuracy)
```

제공된 코드

- **model4student.py : baseline 코드**

```python
import numpy as np
import tensorflow as tf
import tensorflow.contrib.rnn as rnn_cell

def batch_data(shuffled_idx, batch_size, data, labels, start_idx):
    idx = shuffled_idx[start_idx:start_idx+batch_size]
    data_shuffle = [data[i] for i in idx]
    labels_shuffle = [labels[i] for i in idx]

    return np.asarray(data_shuffle), np.asarray(labels_shuffle)

def get_vocabulary_size(X):
    return max([max(x) for x in X]) + 1 # plus the 0th word

def fit_in_vocabulary(X, voc_size):
    return [[w for w in x if w < voc_size] for x in X]

def zero_pad(X, seq_len):
    return np.array([x[:seq_len - 1] + [0] * max(seq_len - len(x), 1) for x in X])

def build_classifier(x, vocabulary_size, EMBEDDING_DIM, HIDDEN_SIZE):
    # Embedding layer
    embeddings_var = tf.Variable(tf.random_uniform([vocabulary_size, EMBEDDING_DIM], -1.0, 1.0), trainable=True)
    batch_embedded = tf.nn.embedding_lookup(embeddings_var, x)

    # RNN layer
    rnn_outputs, states = tf.nn.dynamic_rnn(rnn_cell.BasicRNNCell(HIDDEN_SIZE), batch_embedded, dtype=tf.float32)

    # Fully connected layer
    W = tf.Variable(tf.random_uniform([HIDDEN_SIZE, 2], -1.0, 1.0), trainable=True)
    b = tf.Variable(tf.random_uniform([2], -1.0, 1.0), trainable=True)
    logits = tf.nn.bias_add(tf.matmul(states, W), b)
    hypothesis = tf.nn.softmax(logits)

    return hypothesis, logits

ckpt_path = "output/"

SEQUENCE_LENGTH = 50
EMBEDDING_DIM = 100
HIDDEN_SIZE = 150
BATCH_SIZE = 256
NUM_EPOCHS = 10
learning_rate = 0.001
```

제공된 코드

- **model4student.py : baseline 코드**

```
# Load the data set
np_load_old = np.load

# modify the default parameters of np.load
np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)


x_train = np.load("data/x_train.npy")
y_train = np.load("data/y_train.npy")
x_test = np.load("data/x_test.npy")

np.load = np_load_old


dev_num = len(x_train) // 4

x_dev = x_train[:dev_num]
y_dev = y_train[:dev_num]

x_train = x_train[dev_num:]
y_train = y_train[dev_num:]

y_train_one_hot = tf.squeeze(tf.one_hot(y_train, 2))
y_dev_one_hot = tf.squeeze(tf.one_hot(y_dev, 2))


# Sequences pre-processing
vocabulary_size = get_vocabulary_size(x_train)
x_dev = fit_in_vocabulary(x_dev, vocabulary_size)
x_train = zero_pad(x_train, SEQUENCE_LENGTH)
x_dev = zero_pad(x_dev, SEQUENCE_LENGTH)

batch_ph = tf.placeholder(tf.int32, [None, SEQUENCE_LENGTH], name='batch_ph')
target_ph = tf.placeholder(tf.float32, [None, 2], name='target_ph')


y_pred, logits = build_classifier(batch_ph, vocabulary_size, EMBEDDING_DIM, HIDDEN_SIZE)

loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=target_ph, logits=logits))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)

# Accuracy metric
is_correct = tf.equal(tf.argmax(y_pred, 1), tf.argmax(target_ph, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))

total_batch = int(len(x_train)/BATCH_SIZE) if len(x_train)%BATCH_SIZE == 0 else int(len(x_train)/BATCH_SIZE) + 1
```

제공된 코드

- **model4student.py : baseline 코드**

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    print("학습시작")

    for epoch in range(NUM_EPOCHS):
        print("Epoch", epoch + 1)
        start = 0
        shuffled_idx = np.arange(0, len(x_train))
        np.random.shuffle(shuffled_idx)

        for i in range(total_batch):
            batch = batch_data(shuffled_idx, BATCH_SIZE, x_train, y_train_one_hot.eval(), i * BATCH_SIZE)
            sess.run(optimizer, feed_dict={batch_ph: batch[0], target_ph: batch[1]})
        saver = tf.train.Saver()
        saver.save(sess, ckpt_path)
        saver.restore(sess, ckpt_path)

    dev_accuracy = accuracy.eval(feed_dict={batch_ph: x_dev, target_ph: np.asarray(y_dev_one_hot.eval())})
    print("dev 데이터 Accuracy: %f" % dev_accuracy)

    # 밑에는 건드리지 마세요
    x_test = fit_in_vocabulary(x_test, vocabulary_size)
    x_test = zero_pad(x_test, SEQUENCE_LENGTH)

    test_logits = y_pred.eval(feed_dict={batch_ph: x_test})
    np.save("result", test_logits)
```

점수 산출

- **코드 (70%)**
  - **파일 이름**
    - model4student.py (파이썬 파일 **한 개 로만 작동** 가능 하도록)
  - **Test**
    - 작동 여부를 평가하여 점수 산출
- **보고서 (30%)**
  - **코드 설명**
    - 모델(코드)에 대한 **설명** 명시. (15%)
  - **실험결과**
    - 제출한 result.npy을 통해 모델의 **성능** 비교 (15%)

과제 조건

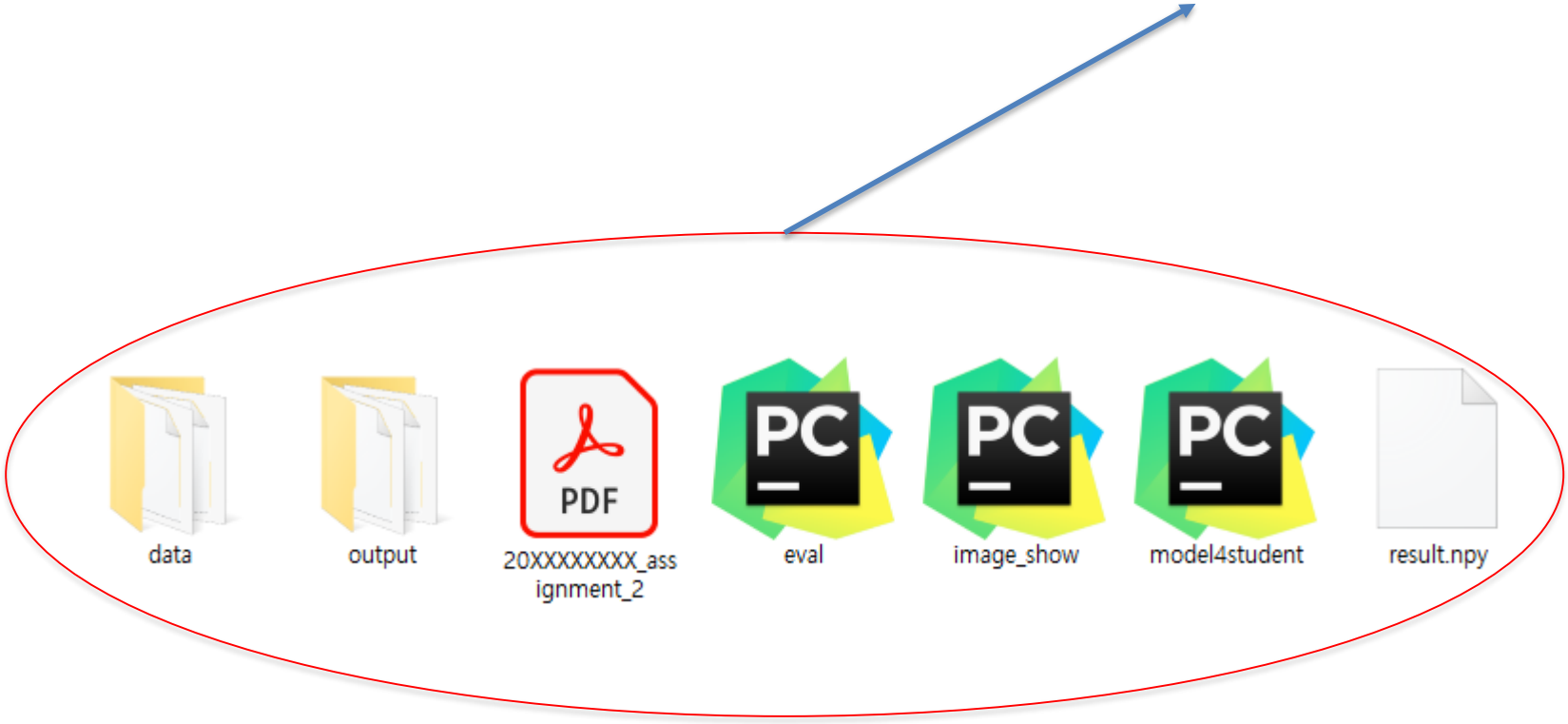
- **환경**
  - 프로그래밍 언어 : **Python 3.7, tensorflow 1.13.1버전**
  - OS : **Windows**
  - 보고서 : **PDF**
- **제출 사항**
  - 파이썬 파일 : **model4student.py**
  - 결과 보고서 : **본인학번_assignment_2.pdf**
  - 결과파일 : **result.npy, output 폴더**
  - 기타 기본 제공 파일 : **eval.py, image_show.py, data 폴더**

주의 사항

- **파일명** 반드시 준수.
- 파일은 **GitLab**에 올려주세요.
- 제출 기한 : **2021.06.04 (23:59)**
- 추가 제출 기한 **없음**.
- 점수 비중 : **코드 70% 보고서 30%**

주의 사항

- **제출하는 파일들을 경로대로 GitLab에 제출!**
- **경로: (GitLab init 경로) – (text classification) – [파일]**



Thank you!