RNN

-Tensorflow

한성대학교
컴퓨터공학과 김바다

CONTENTS.

1. 사전 설정

2. 모델 구성

3. 모델 학습

4. 모델 테스트

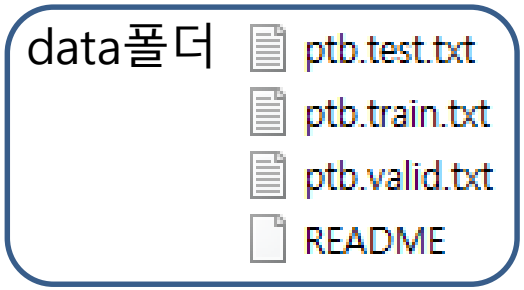
5. 테스트 결과

01. 사전 설정

```
data_path = "./simple-examples/simple-examples/data" # 경로 설정 수정부분

parser = argparse.ArgumentParser()
parser.add_argument('run_opt', type=int, default=1, help='An integer: 1 to train, 2 to test')
parser.add_argument('--data_path', type=str, default=data_path, help='The full path of the tr
args = parser.parse_args()
```

- Data_path 경로 설정



- Run
  - 1 to train (ex $ python rnn.py 1)
  - 2 to test

01. 사전 설정

```
def read_words(filename):
    with tf.gfile.GFile(filename, "r") as f:
        return f.read().decode("utf-8").replace("₩n", "<eos>").split()
```

- 텍스트 파일 단어들을 utf-8형식으로 디코딩.

```
def build_vocab(filename):
  data = read_words(filename)
  counter = collections.Counter(data)
  count_pairs = sorted(counter.items(), key=lambda x: (-x[1], x[0]))
  words, _ = list(zip(*count_pairs))
  word_to_id = dict(zip(words, range(len(words))))
  return word_to_id
```

- 단어는 word : id 쌍으로 묶기

01. 사전 설정

```
def batch_producer(raw_data, batch_size, num_steps):
    raw_data = tf.convert_to_tensor(raw_data, name="raw_data", dtype=tf.int32)

    data_len = tf.size(raw_data)
    batch_len = data_len // batch_size
    data = tf.reshape(raw_data[0: batch_size * batch_len],
                      [batch_size, batch_len])

    epoch_size = (batch_len - 1) // num_steps
```

- **Row데이터 값 셋팅.**
  - Data_len : row데이터 총 길이.
  - Batch_len : 이용가능한 배치 길이를 할당.
  - Data : 배치 사이즈 범위 내로 재구성.
  - Epoch_size : 에폭 반복시 사용될 사이즈

01. 사전 설정

```
def batch_producer(raw_data, batch_size, num_steps):
```

```
    i = tf.train.range_input_producer(epoch_size, shuffle=False).dequeue()
    x = data[:, i * num_steps:(i + 1) * num_steps]
    x.set_shape([batch_size, num_steps])
    y = data[:, i * num_steps + 1: (i + 1) * num_steps + 1]
    y.set_shape([batch_size, num_steps])
    return x, y
```

- 입력 범위 생성자 큐 설정
- x, y 값 셋팅
  - *x* = "A girl walked into a bar, and she"
  - y = "girl walked into a bar, and she said"

02. 모델 구성

```
class Model(object):
    def __init__(self, input, is_training, hidden_size, vocab_size, num_layers,
                 dropout=0.5, init_scale=0.05):
        self.is_training = is_training##기초 값 세팅
        self.input_obj = input
        self.batch_size = input.batch_size
        self.num_steps = input.num_steps
        self.hidden_size = hidden_size
        # create the word embeddings
```

- 기초 값 세팅

```
for d in ['/gpu:0','/gpu:1']: 수정부분
    with tf.device(d):##word embedding creates meaningful vectors to represent each word
        embedding = tf.Variable(tf.random_uniform([vocab_size, self.hidden_size], -init_scale, init_scale))
        inputs = tf.nn.embedding_lookup(embedding, self.input_obj.input_data)##creates look up table
```

- 멀티 GPU사용 셋팅

02. 모델 구성

```
if is_training and dropout < 1:
    inputs = tf.nn.dropout(inputs, dropout)
```

- Drob out구성

```
self.init_state = tf.placeholder(tf.float32, [num_layers, 2, self.batch_size, self.hidden_size])
state_per_layer_list = tf.unstack(self.init_state, axis=0)
rnn_tuple_state = tuple(
    [tf.contrib.rnn.LSTMStateTuple(state_per_layer_list[idx][0], state_per_layer_list[idx][1])
     for idx in range(num_layers)]
)
```

- Placeholder 구성
- LSTM상태 튜플 설정

02. 모델 구성

```
# create an LSTM cell to be unrolled
cell = tf.contrib.rnn.LSTMCell(hidden_size, forget_bias=1.0)
# add a dropout wrapper if training
if is_training and dropout < 1:
    cell = tf.contrib.rnn.DropoutWrapper(cell, output_keep_prob=dropout)
```

- LSTM 셀 구성
- 드롭아웃 적용

```
if num_layers > 1:
    cell = tf.contrib.rnn.MultiRNNCell([cell for _ in range(num_layers)], state_is_tuple=True)
```

- 많은 레이어를 포함해야하기 때문에 멀티RNN셀 적용

02. 모델 구성

```
output, self.state = tf.nn.dynamic_rnn(cell, inputs, dtype=tf.float32, initial_state=rnn_tuple_state)
```

- LSTM셀, embedding vector을 가져와 LSTM네트워크에 로드

```
output = tf.reshape(output, [-1, hidden_size])
    # makes softmax and logistic
softmax_w = tf.Variable(tf.random_uniform([hidden_size, vocab_size], -init_scale, init_scale))
softmax_b = tf.Variable(tf.random_uniform([vocab_size], -init_scale, init_scale))
logits = tf.nn.xw_plus_b(output, softmax_w, softmax_b)
# Reshape logits to be a 3-D tensor for sequence loss
logits = tf.reshape(logits, [self.batch_size, self.num_steps, vocab_size])
```

- Softmax와 logits구성

02. 모델 구성

```
logits = tf.reshape(logits, [self.batch_size, self.num_steps, vocab_size])

# Use the contrib sequence loss and average over the batches
loss = tf.contrib.seq2seq.sequence_loss(# 비용 계산
    logits,
    self.input_obj.targets,
    tf.ones([self.batch_size, self.num_steps], dtype=tf.float32),
    average_across_timesteps=False,
    average_across_batch=True)
# Update the cost
self.cost = tf.reduce_sum(loss)
```

- <Loss 합계를 구하는 방식>
  - 1. 시간 기준
  - 2. 배치 기준

- Loss계산 후 cost에 업데이트

02. 모델 구성

```
self.softmax_out = tf.nn.softmax(tf.reshape(logits, [-1, vocab_size]))# 각단어 출력 0
self.predict = tf.cast(tf.argmax(self.softmax_out, axis=1), tf.int32)#softmax중 가장
correct_prediction = tf.equal(self.predict, tf.reshape(self.input_obj.targets, [-1]))
self.accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))#정확도 평균화
```

- Softmax로 각 단어들 예상확률 집계
- Softmax중 가장 높은 확률의 단어 선별
- 가장 높은 확률 단어와 네트워크 예측을 동일하게 만듦
- 정확도 평균화

02. 모델 구성

```
if not is_training:# 트레이닝중이 아니라면 리턴 시킴
    return
self.learning_rate = tf.Variable(0.0, trainable=False)#학습률 변수

tvars = tf.trainable_variables()
grads, _ = tf.clip_by_global_norm(tf.gradients(self.cost, tvars), 5)
optimizer = tf.train.GradientDescentOptimizer(self.learning_rate)#그
self.train_op = optimizer.apply_gradients(
    zip(grads, tvars),
    global_step=tf.contrib.framework.get_or_create_global_step())
```

- 최적화 작업실시

- GradientDescent 적용

```
self.new_lr = tf.placeholder(tf.float32, shape=[])
self.lr_update = tf.assign(self.learning_rate, self.new_lr)
```

- Learning rate업데이트

03. 모델 학습

```
def train(train_data, vocabulary, num_layers, num_epochs, batch_size, model_save_name,
          learning_rate=1.0, max_lr_epoch=10, lr_decay=0.93):
    # setup data and models
    training_input = Input(batch_size=batch_size, num_steps=30, data=train_data)# 인풋 그
    m = Model(training_input, is_training=True, hidden_size=500, vocab_size=vocabulary,#
              num_layers=num_layers)
    init_op = tf.global_variables_initializer()
```

수정부분

- Input 객체 인스턴스
- Model 객체 인스턴스 생성

03. 모델 학습

```
orig_decay = lr_decay
with tf.Session() as sess:#세션 시작
    # start threads
    sess.run([init_op])
    coord = tf.train.Coordinator()
    threads = tf.train.start_queue_runners(coord=coord)
    saver = tf.train.Saver()
    #변수 초기화 작업 실시
```

- 세션 시작
- 변수 초기화 작업 실시

03. 모델 학습

```
for epoch in range(num_epochs):
    new_lr_decay = orig_decay ** max(epoch + 1 - max_lr_epoch, 0.0)
    m.assign_lr(sess, learning_rate * new_lr_decay)# 매 에폭마다 학습률 계산
    current_state = np.zeros((num_layers, 2, batch_size, m.hidden_size))
    if epoch == 0:
        preAcc = 0
    for step in range(training_input.epoch_size):
        if step % 50 != 0:
            cost, _, current_state = sess.run([m.cost, m.train_op, m.state],
                                              feed_dict={m.init_state: current_state})
        else:#매 50회 반복마다 현재 상태 출력
            cost, _, current_state, acc = sess.run([m.cost, m.train_op, m.state, m.accuracy],
                                                   feed_dict={m.init_state: current_state})
            print("Epoch {}, Step {}, cost: {:.3f}, accuracy: {:.3f}".format(epoch, step, cost, acc))
    # save a model checkpoint
    if acc > preAcc:#현재 에폭이 이번 에폭보다 정확도가 높다면 모델 저장
        print("acc > P-acc:" + str(acc) + " > " + str(preAcc))
        preAcc = acc
        saver.save(sess, data_path + '/' + model_save_name, global_step=epoch)
# do a final save
saver.save(sess, data_path + '/' + model_save_name +'-final')#최종 학습 모델 저장
# close threads
coord.request_stop()
coord.join(threads)
```

- Epoch 반복 작업실시

04. 모델 테스트

```
for batch in range(num_acc_batches):
    if batch == check_batch_idx:
        true_vals, pred, current_state, acc = sess.run([m.input_obj.targets, m.predict, m.state, m.accuracy],
                                                       feed_dict={m.init_state: current_state})
        pred_string = [reversed_dictionary[x] for x in pred[:m.num_steps]]
        true_vals_string = [reversed_dictionary[x] for x in true_vals[0]]
        print("True values (1st line) vs predicted values (2nd line):")
        print("True words".join(true_vals_string))
        print("------------------------------------------------")
        print("Prediced words ".join(pred_string))
    else:
        acc, current_state = sess.run([m.accuracy, m.state], feed_dict={m.init_state: current_state})
    if batch >= acc_check_thresh:
        accuracy += acc
print("Average accuracy: {:.3f}".format(accuracy / (num_acc_batches-acc_check_thresh)))
# close threads
```

- 예상 word와 정답 word
- 정확도

04. 모델 테스트

```
if args.run_opt == 1:
    train(train_data, vocabulary, num_layers=2, num_epochs=150, batch_size=20,
          model_save_name='two-layer-lstm-medium-config-60-epoch-0p93-lr-decay-10-max-lr')
else:
    trained_model = args.data_path + "/two-layer-lstm-medium-config-60-epoch-0p93-lr-decay-10-max-lr-final"
    test(trained_model, test_data, reversed_dictionary)
```

수정부분

수정부분

- If 구문 트레이닝 시
- Else 구문 테스트 시

04. 테스트 결과

```
Epoch 126, Step 1150, cost: 100.836, accuracy: 0.426
Epoch 126, Step 1200, cost: 112.091, accuracy: 0.373
Epoch 126, Step 1250, cost: 114.954, accuracy: 0.364
Epoch 126, Step 1300, cost: 122.776, accuracy: 0.346
acc > P-acc:0.34571427 > 0.34285715
Epoch 127, Step 0, cost: 134.283, accuracy: 0.307
Epoch 127, Step 50, cost: 121.607, accuracy: 0.334
Epoch 127, Step 100, cost: 112.794, accuracy: 0.354
Epoch 127, Step 150, cost: 130.936, accuracy: 0.287
Epoch 127, Step 200, cost: 126.686, accuracy: 0.321
```

```
True values (1st line) vs predicted values (2nd line):
True words
stock market is headed many traders were afraid to trust stock prices quoted on the big board <eos> the futures
halt was even <unk> by big board floor traders <eos> it <unk> things up said
------------------------------------------------
Predicted words
market market is <unk> for traders say willing to be the <eos> <eos> in the big board <eos> the big exchange was
 <unk> more but the board traders traders and the 's the to to
Average accuracy: 0.284
```

- Accuracy : 0.284

- 낮은 정확도이지만 문장 단어의 경향이 유사

감사합니다