Sequence to Sequence Learning with Neural Networks

Group 3

August 18, 2020

Outline

DNN의 한계와 LSTM

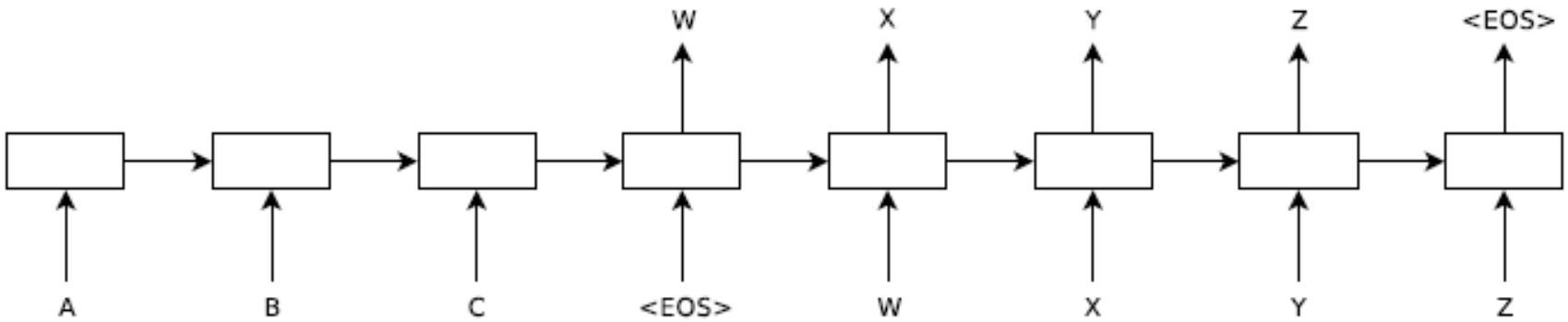
Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priori.

Sequences pose a challenge for DNNs because they require that the dimensionality of the inputs and outputs is known and fixed. In this paper, we show that a straightforward application of the Long Short-Term Memory (LSTM) architecture [16] can solve general sequence to sequence problems. The idea is to use one LSTM to read the input sequence, one timestep at a time, to obtain large fixed-dimensional vector representation, and then to use another LSTM to extract the output sequence from that vector (fig. 1). The second LSTM is essentially a recurrent neural network language model [28, 23, 30] except that it is conditioned on the input sequence. The LSTM's ability to successfully learn on data with long range temporal dependencies makes it a natural choice for this application due to the considerable time lag between the inputs and their corresponding outputs (fig. 1).

DNN(Deep Neural Network)은 유연함과 강력함에도 불구하고, 입력의 차원과 출력의 차원이 이미 정의된 상태에서만 학습이 가능.

하지만 말이나 번역 등은 입력과 출력이 고정적이지 못함. 이를 해결하고 효과적인 학습을 위해 본 논문은 LSTM 활용

모델의 차별점



1. Multi-layered LSTM 모델 사용
1) Encoder : Input sentence를 vector로 만드는데 사용(context vector)
2) Decoder : Encoder로부터 나온 encoded vector를 target sentence로 디코딩

추가적으로,
2. 깊은 LSTM이 얕은 모델보다 더 좋은 성능을 보여 4개의 layer
3. 학습 과정에서 입력의 순서를 뒤집음 input과 target간 관계를 찾기 위한 첫 번째 단어를 찾는 시간이 줄어들게 되며, 이에 따라 backpropagation도 훨씬 수월해짐에 따라 성능이 개선

논문 결과

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

1. Beam size가 클수록, 단어 sequence를 역순으로 넣을 때 BLEU score가 높다.
2. 문장의 길이가 35개의 단어를 넘기는 경우, 정확도가 급격하게 하락

(이 문제 해결을 위해 attention 개념 등장)

3. 기존 SMT(Statistical machine translation)과 비교했을 때 전반적으로 성능이 높음

전처리

```
def preprocess_sentence(sent):
    # 단어와 구두점 사이에 공백 만들기
    # Ex) "he is a boy." => "he is a boy ."
    sent = re.sub(r"([?.!,¿])", r" \1", sent)

    # (a-z, A-Z, ".", "?", "!", ",") 이들을 제외하고는 전부 공백으로 변환
    sent = re.sub(r"[^a-zA-Zㄱ-ㅎㅏ-ㅣ가-힣!.?]+", r" ", sent)
    sent = re.sub(r"\s+", " ", sent)
    return sent
```

```
def load_preprocessed_data(df):
    encoder_input, decoder_input, decoder_target = [], [], []
    for i in range(len(df)):
      src_line = df.iloc[i][0].strip()
      tar_line = df.iloc[i][1].strip()

      # source 데이터 전처리
      src_line_input = [w for w in preprocess_sentence(src_line).split()]

      # target 데이터 전처리
      tar_line = preprocess_sentence(tar_line)
      tar_line_input = [w for w in ("<sos> " + tar_line).split()]
      tar_line_target = [w for w in (tar_line + " <eos>").split()]

      encoder_input.append(src_line_input)
      decoder_input.append(tar_line_input)
      decoder_target.append(tar_line_target)

    return encoder_input, decoder_input, decoder_target
```

1. 단어와 구두점 사이 공백 만들고, 불필요한 문자들 공백 변환

2. 훈련 과정에서 **teacher forcing**을 사용하므로, 훈련 시 사용할 **decoder input** 시퀀스와 실제 값에 해당하는 **decoder target**을 분리하여 저장

토큰화 및 패딩

```
# 인코딩
tokenizer_kor = Tokenizer()
tokenizer_kor.fit_on_texts(sents_kor_in)
encoder_input_train = tokenizer_kor.texts_to_sequences(sents_kor_in)

tokenizer_eng = Tokenizer(filters="", lower=True)
tokenizer_eng.fit_on_texts(sents_eng_in)
tokenizer_eng.fit_on_texts(sents_eng_out)
decoder_input_train = tokenizer_eng.texts_to_sequences(sents_eng_in)
decoder_target_train = tokenizer_eng.texts_to_sequences(sents_eng_out)

# 패딩
encoder_input_train = pad_sequences(encoder_input_train, padding="post")
decoder_input_train = pad_sequences(decoder_input_train, padding="post")
decoder_target_train = pad_sequences(decoder_target_train, padding="post")
```

```
train_df, val_df, test_df = news_df.iloc[:50000, 1:], news_df.iloc[50000:63000, 1:],
news_df.iloc[63000:, 1:]
```

LSTM 모델 설명

1. Keras 사용 -> Teacher forcing 사용함

2. 4 embedded layer -> 2 embedded layer
(구현하는데 어려움이 있었음)

3. batch size 128

4. latent dim : 1000 -> 100
(소요 시간 문제)

5. 순서를 뒤집지 않음
(한국어와 영어의 어순이 달라 상관 없을거라 판단)

6. optimizer SGD -> adam

LSTM 모델 및 결과

```
latent_dim = 100
# 인코더 Layer 2개
encoder_inputs = Input(shape=(None,))
enc_emb =  Embedding(src_vocab_size, latent_dim)(encoder_inputs) # 임베딩 층
enc_masking = Masking(mask_value=0.0)(enc_emb) # 패딩 0은 연산에서 제외

#MultiLayer LSTM
e_outputs, h1, c1 = LSTM(latent_dim, return_state=True, return_sequences=True)
(enc_masking) # 은닉 상태와 셀 상태를 리턴 (output, hidden_state, cell_state)
encoder_lstm= LSTM(latent_dim, return_state=True, return_sequences=True) #모델에 넣어줌
_, h2, c2 = encoder_lstm(e_outputs) # 은닉 상태와 셀 상태를 리턴
encoder_states = [h1, c1, h2, c2] # 인코더의 은닉 상태와 셀 상태를 저장
```

```
# 디코더 Layer 2개
decoder_inputs = Input(shape=(None,))
dec_emb_layer = Embedding(tar_vocab_size, latent_dim) # 임베딩 층
dec_emb = dec_emb_layer(decoder_inputs)
dec_masking = Masking(mask_value=0.0)(dec_emb) # 패딩 0은 연산에서 제외

# 상태값 리턴을 위해 return_state는 True, 모든 시점에 대해서 단어를 예측하기 위해
return_sequences는 True
out_layer1 = LSTM(latent_dim, return_sequences=True, return_state=True)
d_outputs, dh1, dc1 = out_layer1(dec_masking,initial_state= [h2, c2])
# 인코더의 은닉 상태를 초기 은닉 상태(initial_state)로 사용
out_layer2 = LSTM(latent_dim, return_sequences=True, return_state=True)
final, dh2, dc2 = out_layer2(d_outputs, initial_state= [dh1, dc1])

# 모든 시점의 결과에 대해서 소프트맥스 함수를 사용한 출력층을 통해 단어 예측
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(final)

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

LSTM 모델 및 결과

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc'])

checkpoint_dir = './ckpt5'
if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)
callbacks = [tf.keras.callbacks.EarlyStopping(monitor='val_acc', patience=0),
             tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_dir + '/ckpt2-loss=
{loss:.4f}')]
history = model.fit(x = [encoder_input_train, decoder_input_train], y =
decoder_target_train, \
          validation_data = ([encoder_input_test, decoder_input_test],
decoder_target_test),
          batch_size = 128, epochs = 20, callbacks = callbacks)
```

```
Epoch 12/20
528/528 [==============================] - ETA: 0s - loss: 1.8050 - acc: 0.6785INFO:tensorflow:Assets written to: ./ckpt5/ckpt2-loss=1.8050/assets
528/528 [==============================] - 130s 246ms/step - loss: 1.8050 - acc: 0.6785 - val_loss: 2.4307 - val_acc: 0.6137
```

번역기

```
# 인코더 모델
encoder_model = Model(encoder_inputs, encoder_states)

# 디코더 모델 인풋
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_state_input_h1 = Input(shape=(latent_dim,))
decoder_state_input_c1 = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c,
                         decoder_state_input_h1, decoder_state_input_c1]
# 임베딩
dec_emb2= dec_emb_layer(decoder_inputs)

#디코더 모델
d_o, state_h, state_c = out_layer1(dec_emb2, initial_state=decoder_states_inputs[:2])
d_o, state_h1, state_c1 = out_layer2(d_o, initial_state=decoder_states_inputs[-2:])
decoder_states = [state_h, state_c, state_h1, state_c1]
decoder_outputs = decoder_dense(d_o)
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_outputs] + decoder_states)
```

번역기 결과

원문 : 나는 오늘 자정에 한국으로 돌아 가요 .

번역문 : i m going back to korea today at midnight .

예측문 : s s s s there there would there there

원문 : 지금 잠을 자면 깨어나지 못할 거 같아서 지금 가요 .

번역문 : if i fall asleep i might not get up so i will go right now .

예측문 : s s s s s there there would there

원문 : 어제 밤에 왔고 오늘 밤에 가요 .

번역문 : i came yesterday and i will leave today .

예측문 : s s s s s there there would there there

원문 : 다음주 목요일 일에 한국으로 돌아 가요 .

번역문 : i will be going back to korea next thursday .

예측문 : s s s s there there

원문 : 그러나 인보이스의 단가는 잘못된 것 같아 .

번역문 : but i think the price of invoice is wrong .

예측문 : s s s s there there there

BLEU

100문장 으로 테스트 시

Bleu score 0.05

LSTM 모델 및 결과

```
latent_dim = 100
# 인코더 Layer 2개
encoder_inputs = Input(shape=(None,))
enc_emb =  Embedding(src_vocab_size, latent_dim)(encoder_inputs) # 임베딩 층
enc_masking = Masking(mask_value=0.0)(enc_emb) # 패딩 0은 연산에서 제외

#MultiLayer LSTM
e_outputs, h1, c1 = LSTM(latent_dim, return_state=True, return_sequences=True)
(enc_masking) # 은닉 상태와 셀 상태를 리턴 (output, hidden_state, cell_state)
encoder_lstm= LSTM(latent_dim, return_state=True, return_sequences=True) #모델에 넣어줌
_, h2, c2 = encoder_lstm(e_outputs) # 은닉 상태와 셀 상태를 리턴
encoder_states = [h1, c1, h2, c2] # 인코더의 은닉 상태와 셀 상태를 저장

# 디코더 Layer 2개
decoder_inputs = Input(shape=(None,))
dec_emb_layer = Embedding(tar_vocab_size, latent_dim) # 임베딩 층
dec_emb = dec_emb_layer(decoder_inputs)
dec_masking = Masking(mask_value=0.0)(dec_emb) # 패딩 0은 연산에서 제외

out_layer1 = LSTM(latent_dim, return_sequences=True, return_state=True)
d_outputs, dh1, dc1 = out_layer1(dec_masking,initial_state= [h2, c2])
# 인코더의 은닉 상태를 초기 은닉 상태(initial_state)로 사용
out_layer2 = LSTM(latent_dim, return_sequences=True, return_state=True)
final, dh2, dc2 = out_layer2(d_outputs, initial_state= [h1, c1])

decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(final)
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

LSTM 모델 및 결과

원문 : 나는 오늘 자정에 한국으로 돌아 가요 .

번역문 : i m going back to korea today at midnight .

예측문 : you come to korea .

원문 : 지금 잠을 자면 깨어나지 못할 거 같아서 지금 가요 .

번역문 : if i fall asleep i might not get up so i will go right now .

예측문 : you have to be a friend i m not good to me .

원문 : 어제 밤에 왔고 오늘 밤에 가요 .

번역문 : i came yesterday and i will leave today .

예측문 : my friend i work .

원문 : 다음주 목요일 일에 한국으로 돌아 가요 .

번역문 : i will be going back to korea next thursday .

예측문 : you come to korea .

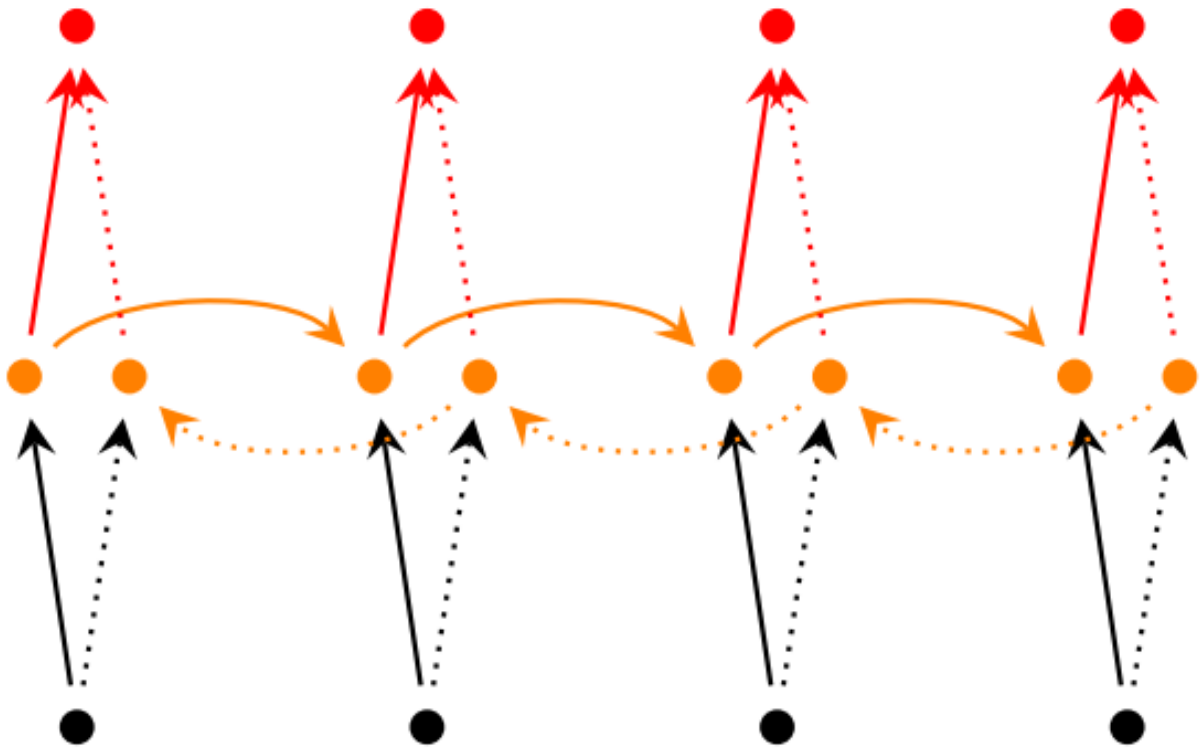
BLEU

12000개 지금 돌리는 중.

10개만 했을 때는 0.178

6400개 했을 때 0.2702554044922252

Bidirectional LSTM 모델 - 모델 레이어 설명



Bidirectional LSTM with attention 모델 설명

1. Keras 사용 -> Teacher forcing 사용함
2. 4 LSTM layer -> 1 bi LSTM layer
3. batch size 128 -> 64
4. latent dim : 1000 -> 256
5. 순서를 뒤집지 않음
6. optimizer SGD($learningrate = 0.75$) -> adam

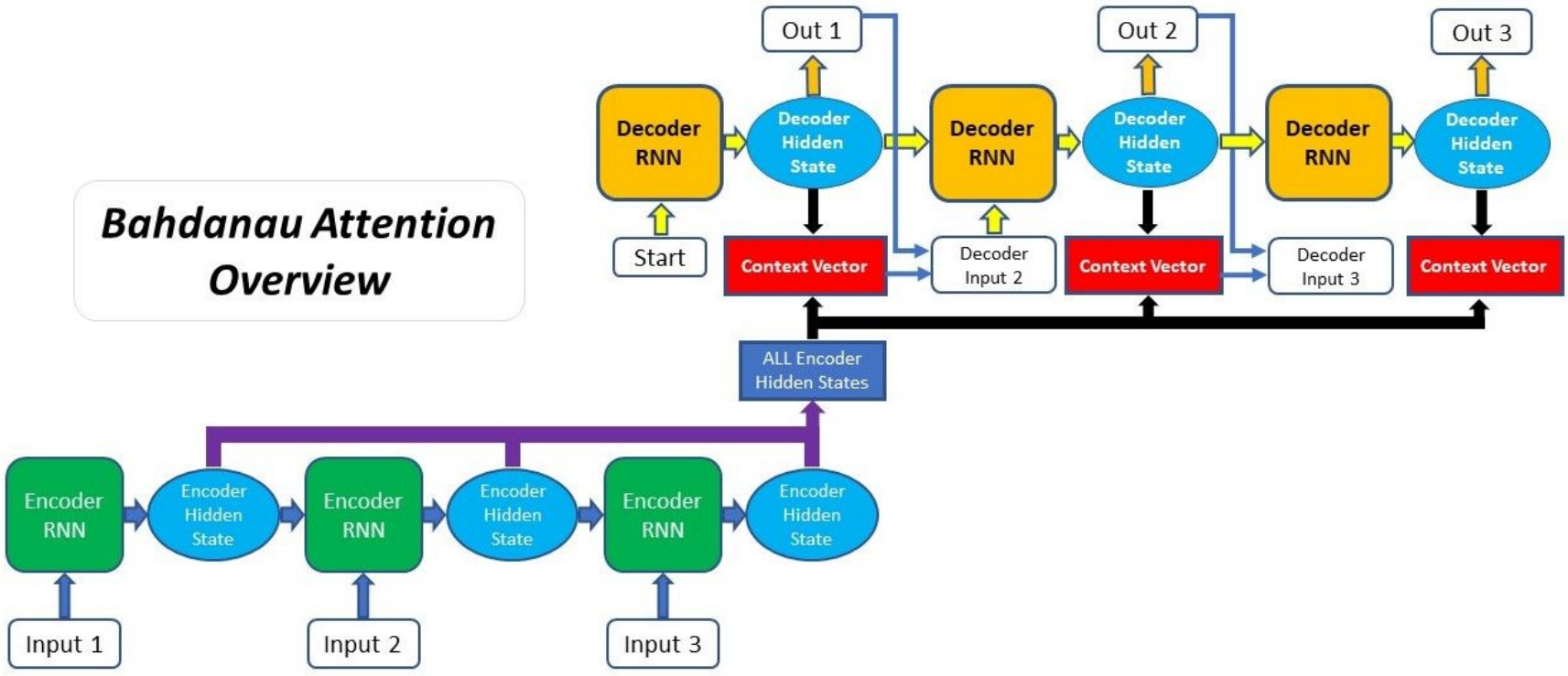
(성능, 예측 면에서 큰 차이를 보임. sgd는 loss 25.0151, adam은 0.6167)

Bidirectional LSTM 모델 - Encoder

```
BUFFER_SIZE = len(input_tensor_train)
BATCH_SIZE = 64
steps_per_epoch = len(input_tensor_train)//BATCH_SIZE
embedding_dim = 256
units = 1024
vocab_inp_size = len(inp_lang.word_index)+1
vocab_tar_size = len(targ_lang.word_index)+1

dataset = tf.data.Dataset.from_tensor_slices((input_tensor_train,
target_tensor_train)).shuffle(BUFFER_SIZE)
dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
```

```
class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
        super(Encoder, self).__init__()
        self.batch_sz = batch_sz
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.bilstm = tf.keras.layers.Bidirectional(LSTM(self.enc_units,
                                        return_sequences=True,

return_state=True,recurrent_initializer='glorot_uniform'))

    def call(self, x, hidden):
        x = self.embedding(x)
        output, state_fh, state_fc, state_bh, state_bc = self.bilstm(x, initial_state =
hidden)
        return output, state_fh, state_fc, state_bh, state_bc

    def initialize_hidden_state(self):
        return [tf.zeros((self.batch_sz, self.enc_units)) for i in range(4)]


encoder = Encoder(vocab_inp_size, embedding_dim, units, BATCH_SIZE)

encoder_hidden = encoder.initialize_hidden_state()
```

Bidirectional LSTM 모델 - attention



Bidirectional LSTM 모델 - attention

```
class BahdanauAttention(tf.keras.layers.Layer):
    def __init__(self, units):
        super(BahdanauAttention, self).__init__()
        self.W1 = tf.keras.layers.Dense(units)
        self.W2 = tf.keras.layers.Dense(units)
        self.V = tf.keras.layers.Dense(1)

    def call(self, query1, query2, values):
        query = Concatenate()([query1, query2])
        query_with_time_axis = tf.expand_dims(query, 1)
        score = self.V(tf.nn.tanh(
            self.W1(query_with_time_axis) + self.W2(values)))

        attention_weights = tf.nn.softmax(score, axis=1)
        context_vector = attention_weights * values
        context_vector = tf.reduce_sum(context_vector, axis=1)

        return context_vector, attention_weights

attention_layer = BahdanauAttention(10)
attention_result, attention_weights = attention_layer(encoder_fh, encoder_sh,
encoder_output)
```

Bidirectional LSTM 모델 - Decoder

```
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.bilstm = tf.keras.layers.Bidirectional(LSTM(self.dec_units,
                                        return_sequences=True,
                                        return_state=True,
                                        recurrent_initializer='glorot_uniform'))
        self.fc = tf.keras.layers.Dense(vocab_size)
        self.attention = BahdanauAttention(self.dec_units)

    def call(self, x, fhidden, shidden, fcell, scell, enc_output):
        context_vector, attention_weights = self.attention(fhidden, shidden, enc_output)
        x = self.embedding(x)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        output, state_fh, state_fc, state_sh, state_sc = self.bilstm(x)
        output = tf.reshape(output, (-1, output.shape[2]))

        x = self.fc(output)

        return x, state_fh, state_fc, state_sh, state_sc, attention_weights
```

```
decoder = Decoder(vocab_tar_size, embedding_dim, units, BATCH_SIZE)

decoder_output, _, _,_,_, _ = decoder(tf.random.uniform((BATCH_SIZE, 1)),
                                      encoder_fh, encoder_fc, encoder_sh, encoder_sc,
encoder_output)
```

Bidirectional LSTM 모델 - optimizer 정의 및 훈련

```
optimizer = tf.keras.optimizers.Adam()
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')

def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask

    return tf.reduce_mean(loss_)
```

```
from tqdm import tqdm
EPOCHS = 10

for epoch in tqdm(range(EPOCHS)):
    start = time.time()

    enc_hidden = encoder.initialize_hidden_state()
    total_loss = 0

    for (batch, (inp, targ)) in enumerate(dataset.take(steps_per_epoch)):
        batch_loss = train_step(inp, targ, enc_hidden)
        total_loss += batch_loss
        if batch % 100 == 0:
            print('Epoch {} Batch {} Loss {:.4f}'.format(epoch + 1,
                                                         batch,
                                                         batch_loss.numpy()))
  # saving (checkpoint) the model every 2 epochs
    if (epoch + 1) % 2 == 0:
        checkpoint.save(file_prefix = checkpoint_prefix)
```

```
Epoch 1 Batch 100 Loss 2.9061
Epoch 1 Batch 200 Loss 2.8334
Epoch 1 Batch 300 Loss 2.6764
Epoch 1 Batch 400 Loss 2.7328
Epoch 1 Batch 500 Loss 2.5698
Epoch 1 Batch 600 Loss 2.5548
Epoch 1 Batch 700 Loss 2.6316
Epoch 1 Batch 800 Loss 2.5674
Epoch 1 Batch 900 Loss 2.3601
```

```
Epoch 10 Batch 0 Loss 0.3117
Epoch 10 Batch 100 Loss 0.3603
Epoch 10 Batch 200 Loss 0.3916
Epoch 10 Batch 300 Loss 0.3639
Epoch 10 Batch 400 Loss 0.3696
Epoch 10 Batch 500 Loss 0.3832
Epoch 10 Batch 600 Loss 0.4003
Epoch 10 Batch 700 Loss 0.3779
Epoch 10 Batch 800 Loss 0.3485
Epoch 10 Batch 900 Loss 0.3948
```

Bidirectional LSTM 모델 - 결과

Input: <start> 아빠는 밥 먹었어 ? <end>
Predicted translation: coffee go to see a lunch ? <end>

Input: <start> 하루에 한번 연락하는 게 그렇게 힘들어 ? <end>
Predicted translation: how is so tired on while ? <end>

Bleu

bleu_score : 0.21738665585704872

예외 : 12000개 중 10035

한계 : OOV에 대한 처리가 필요할듯!

GRU 모델 설명

1. Keras 사용 -> Teacher forcing 사용함

2. 4 lstm layer -> 1 gru layer

3. latent dim : 1000 -> 256

4. 순서를 뒤집지 않음

5. optimizer SGD(learning rate=0.75) -> adam
(성능/예측 면에서 큰 차이를 보임. adam-gru loss는 0.7425)

GRU 결과

```
Input: <start> 몇 시야 ? <en>
Predicted translation: what was the amount of the captain was ?

Input: <start> 나의 고민은 학교가 힘들어 . <end>
Predicted translation: my dinner was my hard to sleep at school . <end>

Input: <start> 나는 학교에 간다 <end>
Predicted translation: i went to the school i go to school . <end>

Input: <start> 아빠는 밥 먹었어 ? <end>
Predicted translation: does the cat has gone with the beer ? <end>
```

BLEU Score

bleu_score : 0.2124

예외: 12000개 중 10035

Mecab 전처리 후 LSTM 모델

```
def pos_sentence(pandas_name, column_name):
  contents_list = pandas_name[column_name].tolist()
#document만 리스트에 넣어두자
  preprocessed_docs = []
  for i in range(len(contents_list)):
    if type(contents_list[i]) != str:
      contents_list[i] = str(contents_list[i])
  preprocessed_contents = []
  for doc in contents_list:
    pos = mecab.pos(doc)
    preprocessed_contents.append(pos)
  all_words = []
  for sentence in preprocessed_contents:
    each_word = []
    for tokens in sentence:
      each_word.append(tokens[0])
    all_words.append(each_word)
  sentence_train = []
  for word in all_words:
    sentence_train.append(' '.join(word))


  return sentence_train
```

```
translate(u'선생 님 이 문장 이 이해 가 안 가 요 .')
```

Input: <start> 선생 님 이 문장 이 이해 가 안 가 요 . <end>

Predicted translation: i want to be a lot of the most popular . <end>

Thank you

곽동명, 김민균, 신재영, 이소연

August 18, 2020