MUD(Meeting Using DeepLearning)
개발 기술 보고서

Today news?(뉴스 요약&추천 서비스)

Team: 박주영(T), 김혜원, 김아연, 홍승환
발표자: 박주영

INDEX

INDEX

Abstractive 요약

(1)Seq2seq model

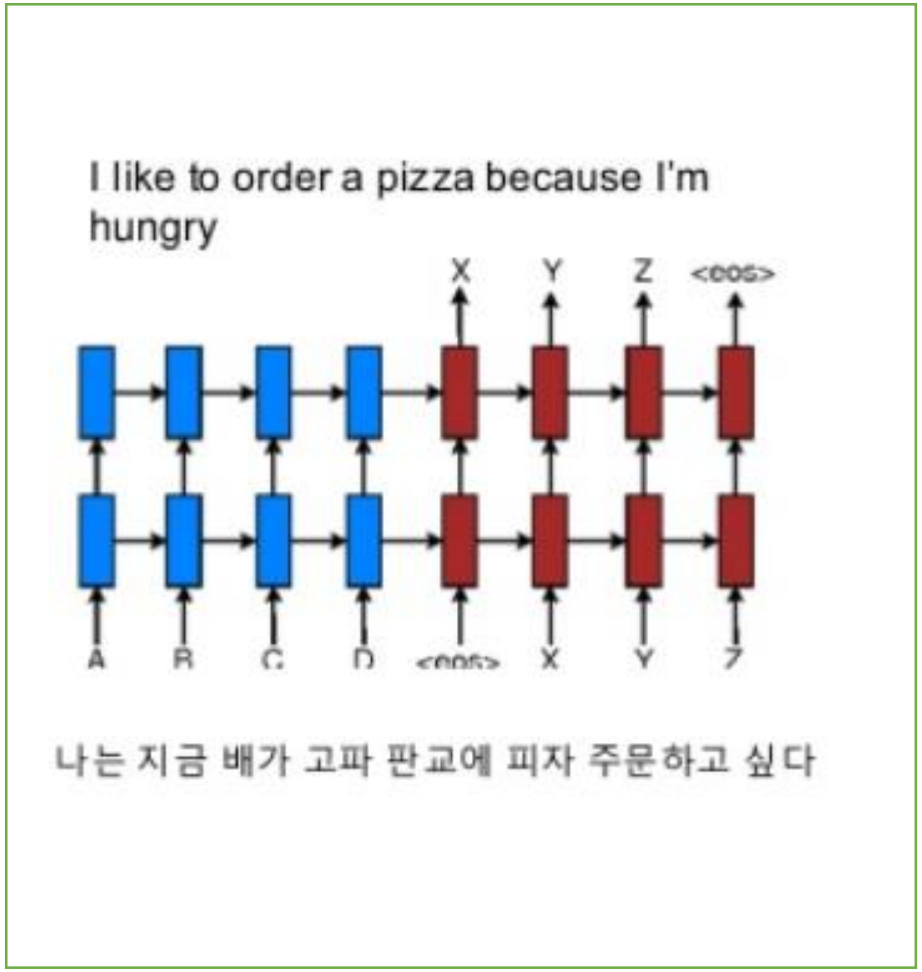
Sequence_to_sequence model은 RNN의 가장 발전된 형태의 아키텍처입니다. LSTM, GRU 등 RNN cell을 길고 깊게 쌓아서 복잡하고 방대한 sequence data를 처리하는데 특화된 모델입니다.

(2) Attention mechanism

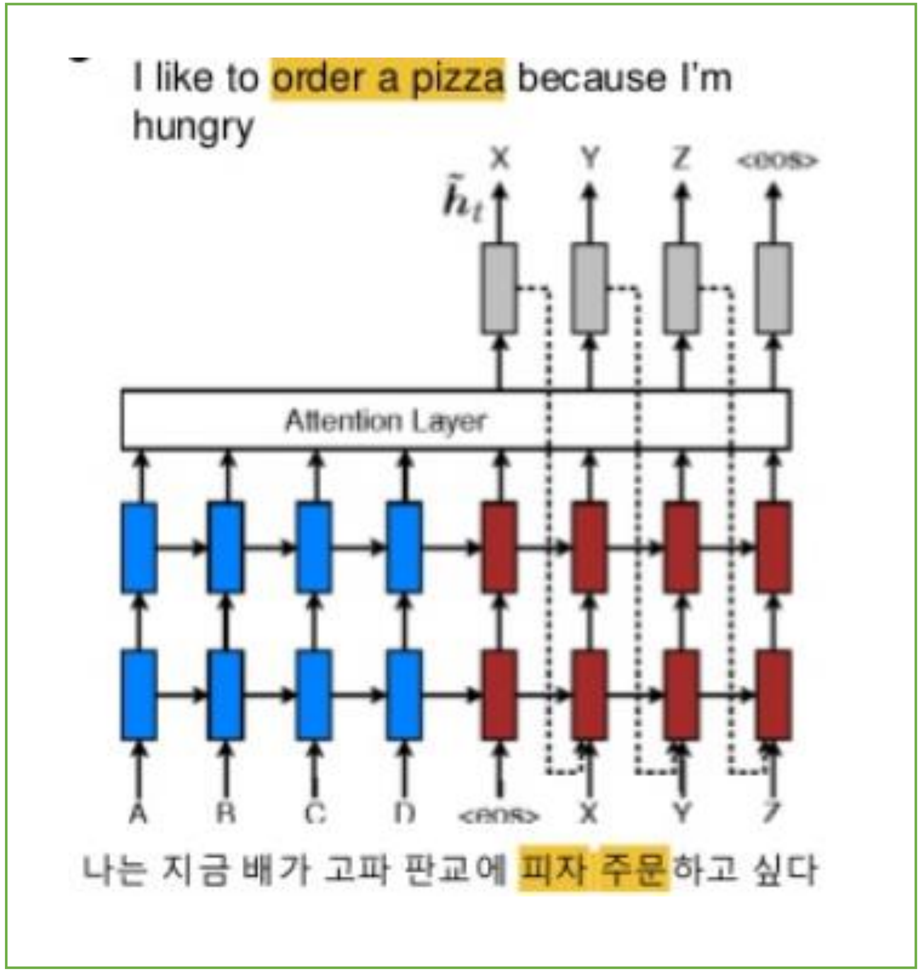
Seq2seq 모델의 가장 큰 문제점은 정보 손실 발생합니다. 이를 위한 대안으로 정확도가 떨어지는 것을 보정해주기 위한 등장한 기법인 어텐션(attention)이 나오게 되었습니다.
Attention mechanism은 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 전체 입력 문장을 다시 한 번 참고한다는 점입니다.

INDEX

Model 구조



Sequence to sequence model



Attention mechanism

순환 신경망

(1)LSTM Cell

- RNN 셀의 장기 의존성 문제를 해결할 뿐만 아니라 학습 또한 빠르게 수렴합니다.
- LSTM은 RNN의 히든 state에 cell-state를 추가한 구조입니다.

(2) GRU Cell

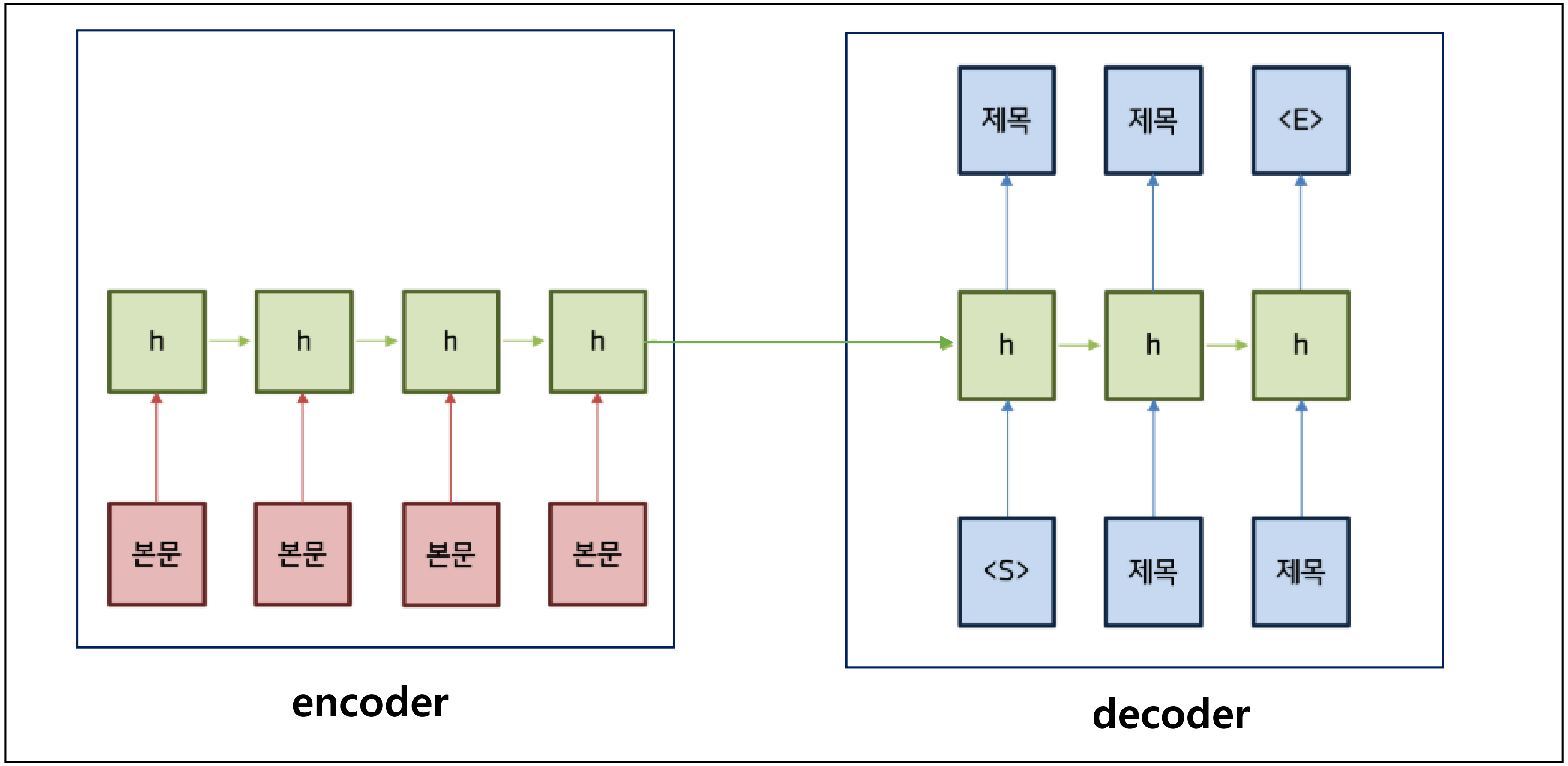
- GRU는 LSTM의 장기 의존성 문제에 대한 해결책을 유지하면서, 은닉 상태를 업데이트하는 계산을 줄였습니다. 다시 말해서, GRU는 성능은 LSTM과 유사하면서 복잡했던 LSTM의 구조를 간단화 시켰습니다.

- LSTM은 출력,입력,삭제 게이트 총 3개의 게이트 존재
GRU는 업데이트 게이트와 리셋 게이트 총 2개의 게이트 존재
GRU는 LSTM보다 학습 속도 빠르다, 하지만 둘의 성능은 비슷하다!

INDEX

2. Test 개요

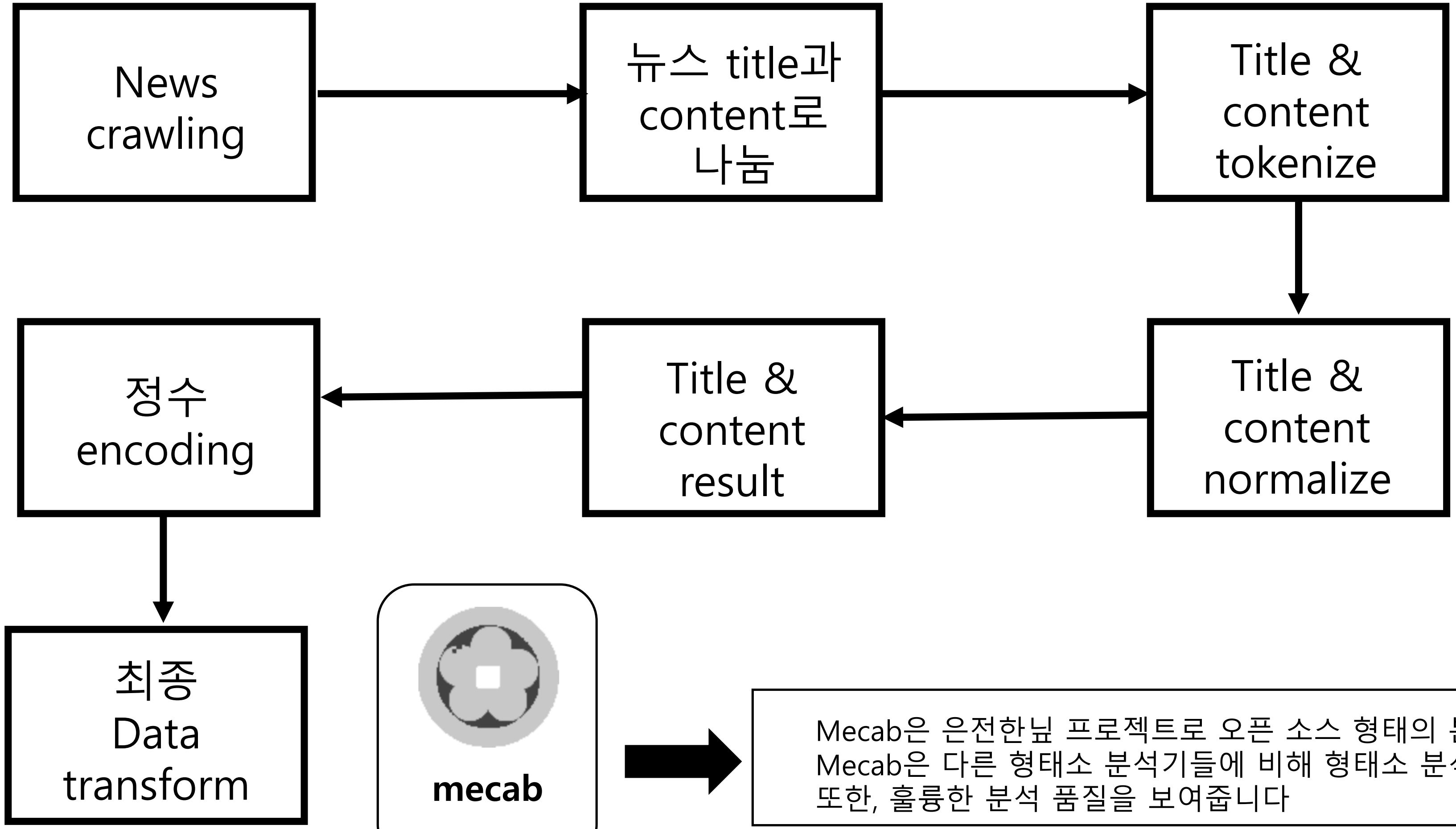
(1) 문제정의



Sequence to sequence model은 정답이 있는 data만 가능하다!

3. Test

(1) Data 전처리 과정



mecab

Mecab은 은전한닢 프로젝트로 오픈 소스 형태의 분석기 입니다
Mecab은 다른 형태소 분석기들에 비해 형태소 분석 시간이 빠릅니다.
또한, 훌륭한 분석 품질을 보여줍니다

INDEX

3. Test

(1) Data 전처리 과정

```python
def normalize(datapath):
    #csv file save
    os.chdir("./")
    doc_ko=open('merge11.csv','w',encoding='utf_8_sig', newline="")
    wcsv = csv.writer(doc_ko)

    #mecab tokenize
    m = Mecab()

    #file1 = open('merge10.csv','r',encoding='utf_8_sig')
    file1 = open(datapath,'r',encoding='utf_8_sig')
    line = csv.reader(file1)

    title,content,title1,content1,title2,content2=[],[],[],[],[],[]
    m_title,m_content=[],[]
    count=1
    temp=''
    for i in line:
        hangul = re.compile('[^ ㄱ-ㅣ가-힣]+')
        i[0] = hangul.sub('',i[0])
        i[1] = hangul.sub('',i[1])
        m_title.append(m.morphs(i[0]))
        m_content.append(m.morphs(i[1]))
        #stop word delete
        for w in m_title:
            if w not in stopwords:
                title.append(w)
        for p in m_content:
            if p not in stopwords:
                content.append(p)

        title1 = ' '.join(title[0])
        title2.append(title1)
        content1 = ' '.join(content[0])
        content2.append(content1)
        temp=content1
        wcsv.writerow([title1,content1])
```

```python
        title1,content1='',''
        title,content,title1,content1,=[],[],[],[]
        m_title,m_content=[],[]

        if count == 10000:
            break
        else:
            count += 1
    len(title2+content2)
    return title2,content2
```

result

```
train
/home/usergpu/PycharmProjects/seq2seq/venv/bin/python /home/usergpu/PycharmProjects/seq2seq/train.py

title:  ['년 만 의 연말 성과급 국민은행 최대 만 원 지급']
content:  ['국민은행 이 올 연말 직원 들 에게 만 만 원 성과급 을 지급 하 기 로 했 다 일 금융 권 에 따르 면 국민은행 은 최근 분기 노사
  협의회 에서 기본급 의 에 해당 하 는 연말 특별 보로금 을 지급 하 기 로 합의 했 다 올해 호 실적 이 확실 시 되 는 상황 이 어서 우선 기
 본급 의 를 지급 하 고 연말 결산 이 끝나 면 내년 초 에 잔여 금 을 주 기 로 한 것 이 다 이 에 따라 국민은행 직원 은 연말 에 만 만 만
 원 의 성과급 을 받 게 됐 다 국민은행 은 년 을 마지막 으로 지난 년 간 연말 특별 성과급 을 지급 하 지 않 았 다 이 는 윤종규 금융 지주
 회장 이 언급 한 초과 이익 배분 제 의 일환 으로 풀이 된다 윤 회장 은 지난 월 임직원 정기 조회 에서 땀 의 결실 인 초과 이익 을 당당 하
 게 공유 할 수 있 도록 할 것 이 라며 지속 가능 한 보상 체계 로 이익 배분 제 를 합리 적 으로 재 정비 하 겠 다 고 밝힌 바 있 다']
contents number: 20000, voca numbers: 8330335

Process finished with exit code 0
```

3. Test

(1) Data 전처리 과정

```python
# word_to_ix, ix_to_word 생성
def make_dict(contents):
    content=[]
    for i in contents:
        for word in i.split():
            content.append(word)
    #print(content[:3])
    vocab=Counter(content)
    #print(vocab)
    vocab = Counter(content)
    maxn = max(vocab.values())
    vocab['<PAD>']=maxn+1
    vocab['<S>'] = maxn + 2
    vocab['<E>'] = maxn + 3
    vocab['<UNK>'] = maxn + 4
    #print(vocab['<PAD>'])
    ix_to_word = {ch: i for i, ch in vocab.items()}

    word_to_ix = vocab
    counte=0
    for i,k in ix_to_word.items():
        if counte == 5:break
        counte += 1
        print("ix_to_word: {",i,": ",k,"}")
    counte=0
    for i,k in word_to_ix.items():
        if counte == 5: break
        counte += 1
        print("word_to_ix: {",i,": ",k,"}")

    print('contents number: %s, voca numbers: %s' %(len(contents),len(ix_to_word)))
    return word_to_ix,ix_to_word
```

result

```
train
/home/usergpu/PycharmProjects/seq2seq/venv/bin/python
ix_to_word: { 1 :  처해진다 }
ix_to_word: { 2 :  두두두두 }
ix_to_word: { 3 :  떨어뜨렸 }
ix_to_word: { 4 :  컴파일 }
ix_to_word: { 5 :  트웰브 }
word_to_ix: { 조용히 :  19 }
word_to_ix: { 비스타 :  7 }
word_to_ix: { 무너뜨리 :  5 }
word_to_ix: { 차릴 :  1 }
word_to_ix: { 클 :  364 }
contents number: 20000, voca numbers: 1320
```

3. Test

(1) Data 전처리 과정

```
def make_batch(encoder_inputs, decoder_inputs, targets, target_weights):
    encoder_size = len(encoder_inputs[0])
    decoder_size = len(decoder_inputs[0])
    encoder_inputs, decoder_inputs, targets, target_weights = \
        np.array(encoder_inputs), np.array(decoder_inputs), np.array(targets), np.array(target_weights)
    result_encoder_inputs = []
    result_decoder_inputs = []
    result_targets = []
    result_target_weights = []
    for i in range(encoder_size):
        result_encoder_inputs.append(encoder_inputs[:, i])
    for j in range(decoder_size):
        result_decoder_inputs.append(decoder_inputs[:, j])
        result_targets.append(targets[:, j])
        result_target_weights.append(target_weights[:, j])
    return result_encoder_inputs, result_decoder_inputs, result_targets, result_target_weights
```

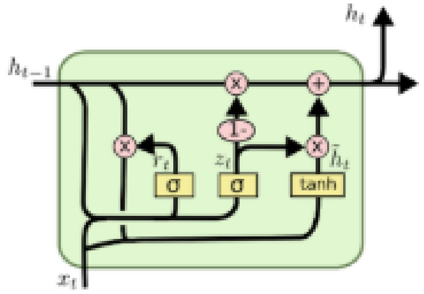
result

```
[2141, 10696, 929, 36, 29919, 1718, 13040, 96, 2387, 2075, 12001, 525, 29919, 943, 38626, 278, 1671, 1, 29919, 1718, 10696, 17009, 52, 34, 29919, 339, 2236, 11, 57263, 17817, 334, 56381, 12662, 2708, 271, 4141, 17817, 3785, 262, 56381, 10577, 409,
[60770, 11, 96, 1718, 29919, 36, 929, 1280, 521, 37061, 170, 34, 22061, 732, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769]
[11, 96, 1718, 29919, 36, 929, 1280, 521, 37061, 170, 34, 22061, 732, 60771, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769, 60769]
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Process finished with exit code 0
```

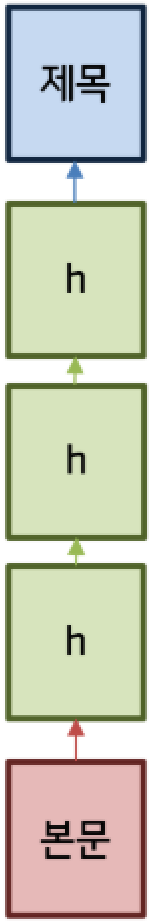
INDEX

3. Test

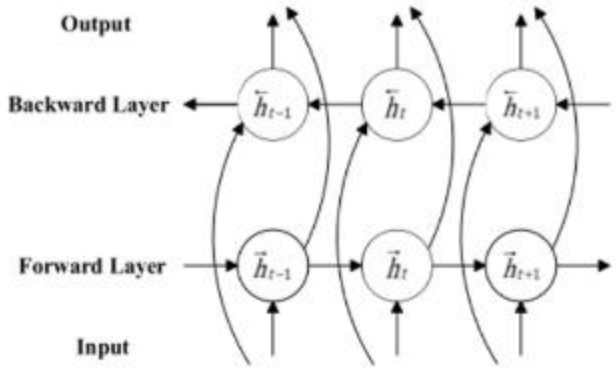
(2)Model 구현



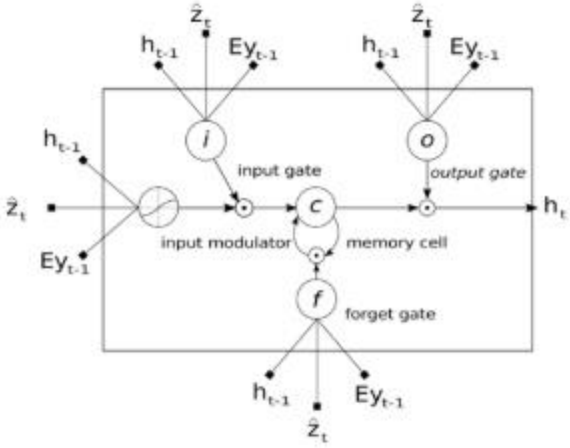
GRU



multi



bidirectional



attention

INDEX

3. Test

(2)Model 구현

- [1] 초기 설정

```
import tensorflow as tf
import numpy as np
import util
import time

datapath = 'mergel1.csv'
title,content = util.normalize(datapath)

contents=title+content

word_to_ix,ix_to_word = util.make_dict(contents)

|
forward_only = False
hidden_size = 300
vocab_size = len(ix_to_word)
num_layers = 3
learning_rate = 0.001
batch_size = 16
encoder_size = 100
decoder_size = util.doclength(title,sep=True) # (Maximum) number of time steps in this batch
steps_per_checkpoint = 10

# transform data
encoderinputs, decoderinputs, targets_, targetweights = util.make_suffle(content,title,word_to_ix,encoder_size=encoder_size,decoder_size=decoder_size,shuffle=False)
```

3. Test

(2)Model 구현

- [1] seq2seq 변수 선언

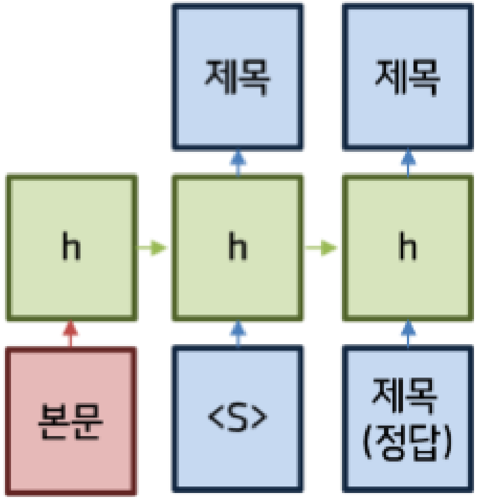
```
# variables
self.source_vocab_size = vocab_size
self.target_vocab_size = vocab_size
self.batch_size = batch_size
self.encoder_size = encoder_size
self.decoder_size = decoder_size
self.learning_rate = tf.Variable(float(learning_rate), trainable=False)
self.global_step = tf.Variable(0, trainable=False)

# networks
W = tf.Variable(tf.random.normal([hidden_size, vocab_size]))
b = tf.Variable(tf.random.normal([vocab_size]))
output_projection = (W, b)
self.encoder_inputs = [tf.compat.v1.placeholder(tf.int32, [batch_size]) for _ in range(encoder_size)] #
self.decoder_inputs = [tf.compat.v1.placeholder(tf.int32, [batch_size]) for _ in range(decoder_size)]
self.targets = [tf.compat.v1.placeholder(tf.int32, [batch_size]) for _ in range(decoder_size)]
self.target_weights = [tf.compat.v1.placeholder(tf.float32, [batch_size]) for _ in range(decoder_size)]
```

- [2] seq2seq network 선언

```
single_cell = tf.compat.v1.nn.rnn_cell.GRUCell(num_units=hidden_size)
cell = tf.compat.v1.nn.rnn_cell.MultiRNNCell([single_cell] * num_layers)
```

3. Test

(2)학습(train)



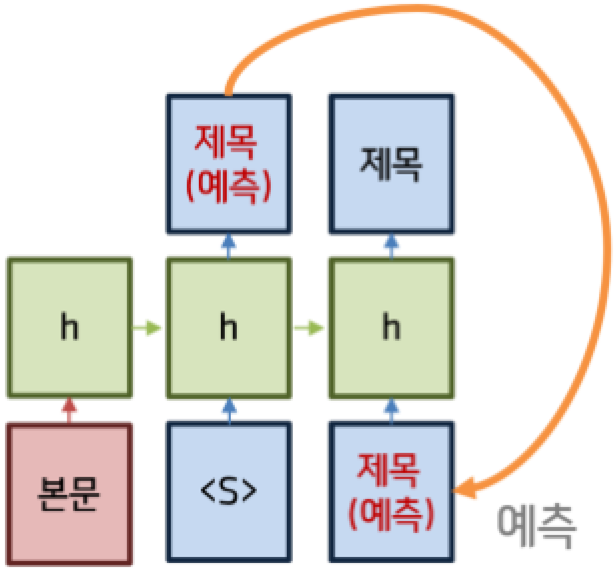
학습

```
self.outputs, self.states = tf.contrib.legacy_seq2seq.embedding_attention_seq2seq(
```

```
if not forward_only:
    self.outputs, self.states = tf.contrib.legacy_seq2seq.embedding_attention_seq2seq(
        self.encoder_inputs, self.decoder_inputs, cell,
        num_encoder_symbols=vocab_size,
        num_decoder_symbols=vocab_size,
        embedding_size=hidden_size,
        output_projection=output_projection,
        feed_previous=False)

    self.logits = [tf.matmul(output, output_projection[0]) + output_projection[1] for output in self.outputs]
    self.loss = []
    for logit, target, target_weight in zip(self.logits, self.targets, self.target_weights):
        crossentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=logit, labels=target)
        self.loss.append(crossentropy * target_weight)
    self.cost = tf.compat.v1.add_n(self.loss)
    self.train_op = tf.compat.v1.train.AdamOptimizer(learning_rate).minimize(self.cost)
```

3. Test

(2)예측(precision,test)

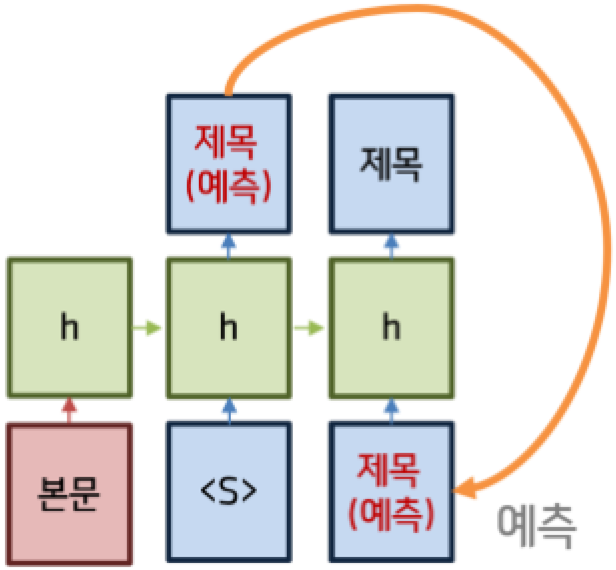



```
output_projection=output_projection,
feed_previous=True)
```

```
else:
    self.outputs, self.states = tf.contrib.legacy_seq2seq.embedding_attention_seq2seq(
        self.encoder_inputs, self.decoder_inputs, cell,
        num_encoder_symbols=vocab_size,
        num_decoder_symbols=vocab_size,
        embedding_size=hidden_size,
        output_projection=output_projection,
        feed_previous=True)
    self.logits = [tf.matmul(output, output_projection[0]) + output_projection[1] for output in self.outputs]
```

3. Test

(2)예측(precision,test)




```
output_projection=output_projection,
feed_previous=True)
```

```
else:
    self.outputs, self.states = tf.contrib.legacy_seq2seq.embedding_attention_seq2seq(
        self.encoder_inputs, self.decoder_inputs, cell,
        num_encoder_symbols=vocab_size,
        num_decoder_symbols=vocab_size,
        embedding_size=hidden_size,
        output_projection=output_projection,
        feed_previous=True)
    self.logits = [tf.matmul(output, output_projection[0]) + output_projection[1] for output in self.outputs]
```

3. Test

- 오류(error)

실제 결과물

```
InvalidArgumentError (see above for traceback): indices[0] = 244392 is not in [0, 1818)
         [[Node: embedding_attention_seq2seq/embedding_attention_decoder/embedding_lookup_29 = Gather[Tindices=DT_INT32, Tparams=DT_FLOAT, _class=["l
bedding_attention_seq2seq/embedding_attention_decoder/embedding"], validate_indices=true, _device="/job:localhost/replica:0/task:0/device:CPU:0"](emb
_attention_seq2seq/embedding_attention_decoder/embedding/read, _arg_Placeholder_129_0_32)]]
```

INDEX

차후 계획

Headline

Headline을 추출하려 할 때, 다중 문서에 대한 headline 추출해야 한다. 하지만 이와 관련 문서가 거의 없다.

정확성

Headline을 attention을 이용하여 추출하려 할 때, 정확성이 많이 떨어진다. 이에 정확성을 올리기 위한 방법 생각해야함.

Data set

최종 목표는 뉴스 본문에 대한 요약본이다.문제는 dataset이 없다.현재 dataset으로는 뉴스 중간에 포함 되어있는 중간 제목들을 이용하여 요약할 계획.

Attention으로 요약이 가능한가?

현재 제가 찾아본 요약 관련 알고리즘은 주로 textrank또는 lexrank입니다.
하지만 seq2seq model에 copy mechanism과 pointer nerwork를 이용한 다른 몇 논문을 참고하여 요약할 예정.

참고 문헌

https://reniew.github.io/31 → **Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation**

https://git.mif.vu.lt/TankBusterPBL/TankBuster/blob/2583045df556e522a1a14fc2de35cc4ec43dd596/bin/Tensorflow/Tensorflow/tutorials/rnn/translate/seq2seq_model.py

https://github.com/dongjun-Lee/text-summarization-tensorflow

https://github.com/graykode/nlp-tutorial/blob/master/4-2.Seq2Seq(Attention)/Seq2Seq(Attention)_Tensor.ipynb

https://wikidocs.net/22893

https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/contrib/legacy_seq2seq/embedding_attention_seq2seq

https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/seq2seq/

https://github.com/petewarden/tensorflow_makefile/blob/master/tensorflow/models/rnn/translate/seq2seq_model.py

→seq2seq과 ATTENTION 기술 설명

https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/
→LSTM

THANK YOU!