NLP Project
-Final Report-

강유정 김근호 김나연
김정현 박진우 임형준

KU-BIG

Index

1. EDA
2. Model
   - Seq2Seq
   - Transformer
3. Project Result
4. Need to complement

KU-BIG

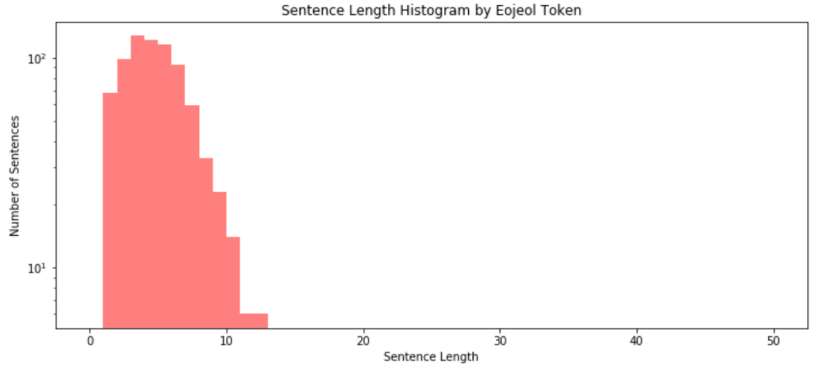
1. EDA (Data)

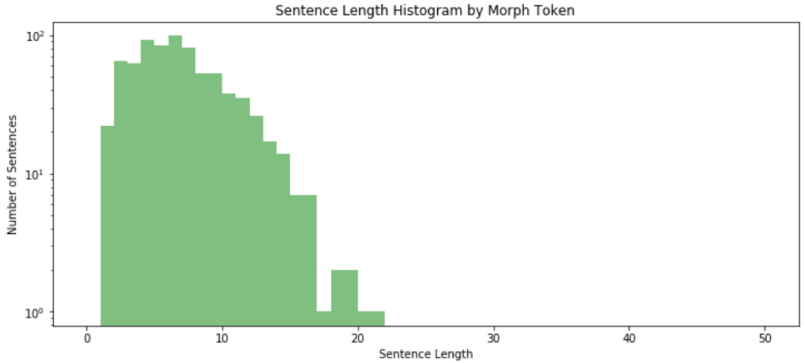
Data Sample 개수: 383

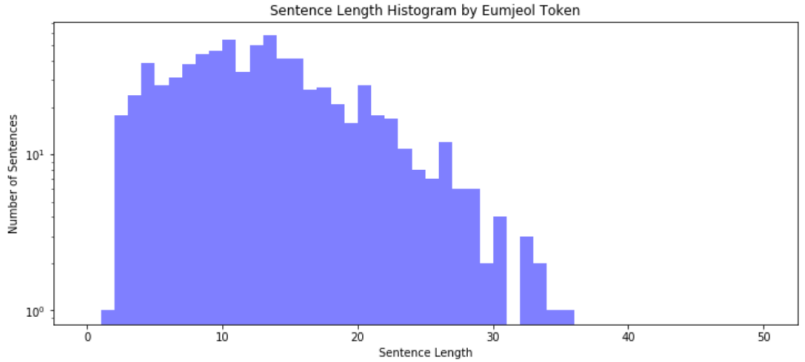
Q	A
여기 니 방 아니구 내 방 맞거든?	누가 뭐래?
너 왜 여기 있어?	내가 볼일 없이 나타나는 거 봤냐?
이거 주려구 온거야?	덮치기라두 할 줄 알았냐?
이상한 일 꾸미는 줄 알았어	난 날 좋아하는 여자가 아님 안 건드려.
내가 애야? 혼자 산책 할 수도 있지	물에 빠졌나 뱀에 물렸나 별 생각이 다 드는데
너두 줄까?	그래
잊어버리기라도 하면 니가 날 가만두	그걸 알면 됐어
괜찮아?	괜찮아
어딜 가는 거야?	가만히 있어봐 보여줄게 있어
하트야! 정말...하트가 있어 저기!	좋아하는 여자가 생기면 데려와야지 결심했었
정말 멋진 풍경이야	보이냐 내 마음?
지후 녀석 비상이다.	무슨 일이야?
오후 내내 안보여. 방에 짐은 그대론데	괜찮을거야 그녀석은 내가 알아
어떻게 들어왔어?	여기 주인이 누군지 잊었냐?
응? 그게 무슨 말이야?	내 마음 접수했냐?
구준표! 오해야 이건	이건 뭐? 이것도 니가 모르는 일이야?
미안해	난 너한테 내 진심을 보여줬어. 그랬는데 니 대
준표야! 오해라니까	닥쳐!

KU-BIG

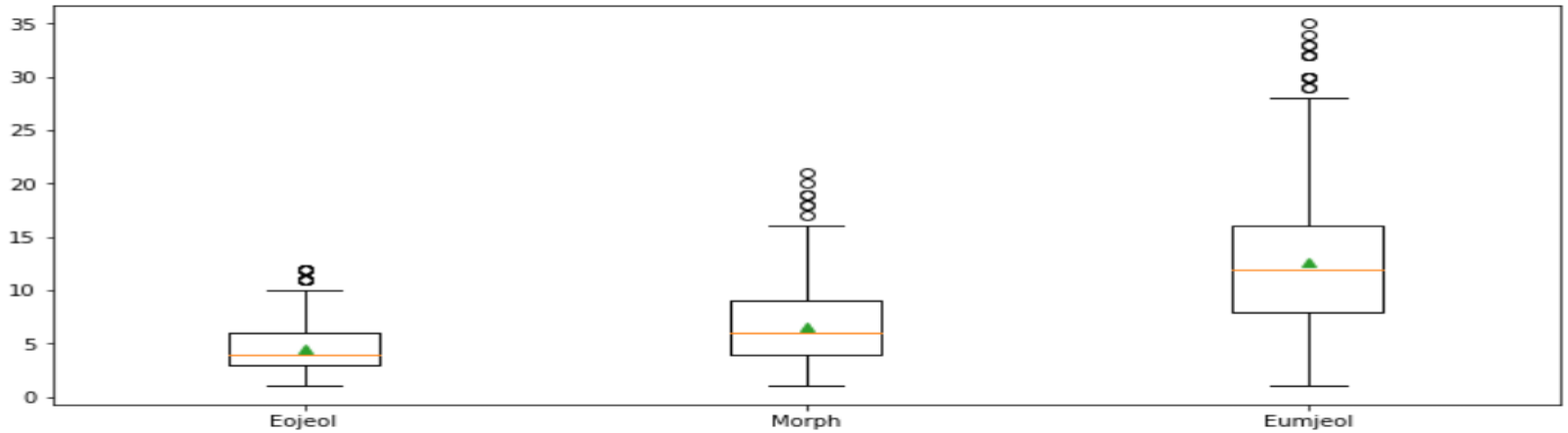
1. EDA (Histogram)







1. EDA (Box Plot)



어절 최대길이 : 12
어절 최소길이 : 1
어절 평균길이 : 4.48
어절 길이 표준편차 : 2.36
어절 중간길이 : 4.0
제 1 사분위 길이 : 3.0
제 3 사분위 길이 : 6.0

형태소 최대길이 : 21
형태소 최소길이 : 1
형태소 평균길이 : 6.59
형태소 길이 표준편차 : 3.56
형태소 중간길이 : 6.0
형태소 1/4 퍼센타일 길이 : 4.0
형태소 3/4 퍼센타일 길이 : 9.0

음절 최대길이 : 35
음절 최소길이 : 1
음절 평균길이 : 12.60
음절 길이 표준편차 : 6.57
음절 중간길이 : 12.0
음절 1/4 퍼센타일 길이 : 8.0
음절 3/4 퍼센타일 길이 : 16.0

KU-BIG

1. EDA (Q&A)

```
print('형태소 최대길이: {}'.format(np.max(query_sent_len_by_morph)))
print('형태소 최소길이: {}'.format(np.min(query_sent_len_by_morph)))
print('형태소 평균길이: {:.2f}'.format(np.mean(query_sent_len_by_morph)))
print('형태소 길이 표준편차: {:.2f}'.format(np.std(query_sent_len_by_morph)))
print('형태소 중간길이: {}'.format(np.median(query_sent_len_by_morph)))
print('형태소 1/4 퍼센타일 길이: {}'.format(np.percentile(query_sent_len_by_morph, 25)))
print('형태소 3/4 퍼센타일 길이: {}'.format(np.percentile(query_sent_len_by_morph, 75)))
```

```
형태소 최대길이: 21
형태소 최소길이: 1
형태소 평균길이: 6.66
형태소 길이 표준편차: 3.49
형태소 중간길이: 6.0
형태소 1/4 퍼센타일 길이: 4.0
형태소 3/4 퍼센타일 길이: 8.0
```

Question Dataset

```
형태소 최대길이: 19
형태소 최소길이: 1
형태소 평균길이: 6.53
형태소 길이 표준편차: 3.64
형태소 중간길이: 6.0
형태소 1/4 퍼센타일 길이: 4.0
형태소 3/4 퍼센타일 길이: 9.0
```

Max Sequence Length = 19

Answer Dataset

KU-BIG

1. EDA (Word Cloud)

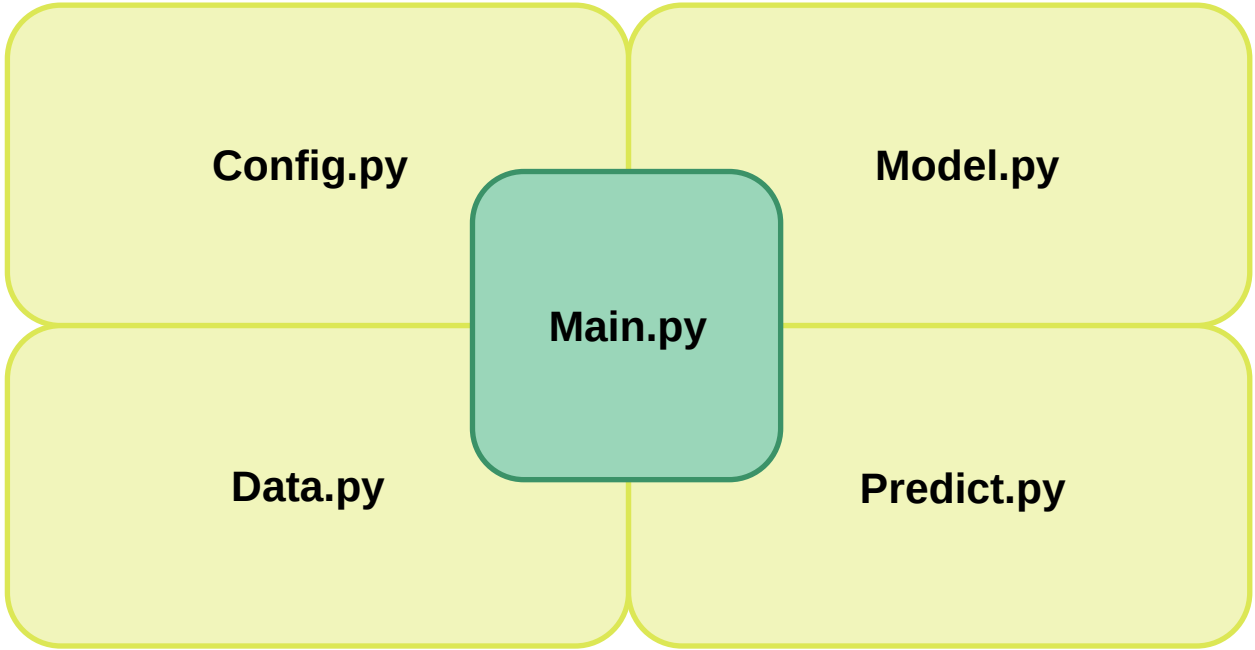
```
query_wordcloud = WordCloud(font_path= DATA_IN_PATH + 'NanumGothic.ttf').generate(answer_NVA_token_sentences)

plt.imshow(query_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Question Dataset

Answer Dataset

2. Model

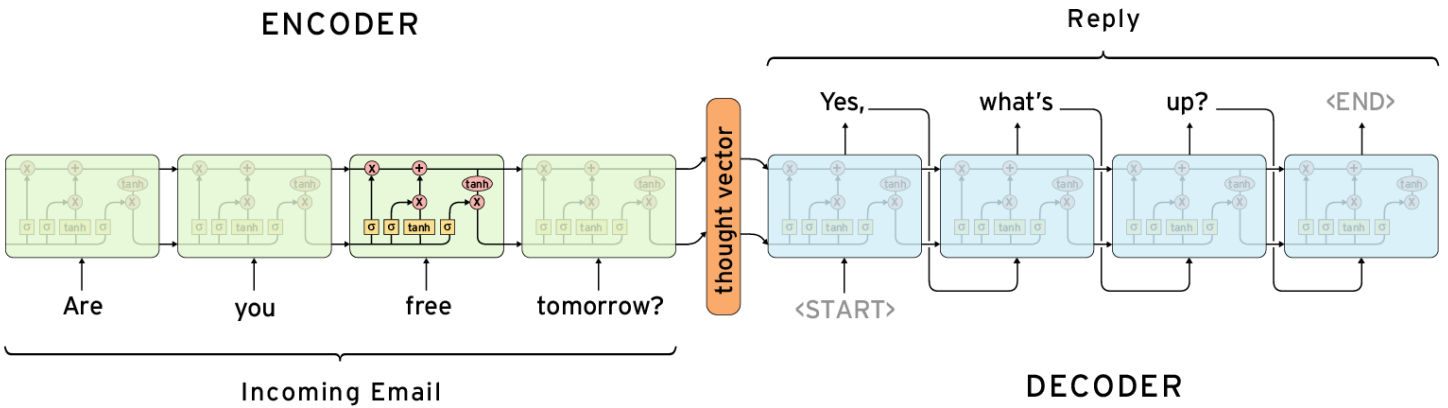


Check List

- ✓ Tokenizing
- ✓ Word Indexing
- ✓ Embedding
- ✓ Making Vocabulary
- ✓ Encoder & Decoder
- ✓ Optimizing

KU-BIG

2. Model – Seq2seq



2. Model – Seq2seq (config.py)

Hyperparameter

```python
import tensorflow as tf

tf.app.flags.DEFINE_integer('batch_size', 64, 'batch size') # 배치 크기
tf.app.flags.DEFINE_integer('train_steps', 20000, 'train steps') # 학습 에포크
tf.app.flags.DEFINE_float('dropout_width', 0.8, 'dropout width') # 드롭아웃 크기
tf.app.flags.DEFINE_integer('layer_size', 1, 'layer size') # 멀티 레이어 크기 (multi rnn)
tf.app.flags.DEFINE_integer('hidden_size', 128, 'weights size') # 가중치 크기
tf.app.flags.DEFINE_float('learning_rate', 1e-3, 'learning rate') # 학습률
tf.app.flags.DEFINE_float('teacher_forcing_rate', 0.7, 'teacher forcing rate') # 학습시 디코더 인풋 정답 지원율
tf.app.flags.DEFINE_string('data_path', '../data_in/ChatBotData.csv', 'data path') #  데이터 위치
tf.app.flags.DEFINE_string('vocabulary_path', './data_out/vocabularyData.voc', 'vocabulary path') # 사전 위치
tf.app.flags.DEFINE_string('check_point_path', './data_out/check_point', 'check point path') # 체크 포인트 위치
tf.app.flags.DEFINE_string('save_model_path', './data_out/model', 'save model') # 모델 저장 경로
tf.app.flags.DEFINE_integer('shuffle_seek', 1000, 'shuffle random seek') # 셔플 시드값
tf.app.flags.DEFINE_integer('max_sequence_length', 19, 'max sequence length') # 시퀀스 길이
tf.app.flags.DEFINE_integer('embedding_size', 128, 'embedding size') # 임베딩 크기
tf.app.flags.DEFINE_boolean('embedding', True, 'Use Embedding flag') # 임베딩 유무 설정
tf.app.flags.DEFINE_boolean('multilayer', True, 'Use Multi RNN Cell') # 멀티 RNN 유무
tf.app.flags.DEFINE_boolean('attention', True, 'Use Attention') #  어텐션 사용 유무
tf.app.flags.DEFINE_boolean('teacher_forcing', True, 'Use Teacher Forcing') # 학습시 디코더 인풋 정답 지원 유무
```


KU-BIG

2. Model – Seq2seq (data.py)

Word Indexing

Tokenizing

```
def prepro_like_morphlized(data):


    morph_analyzer = Okt()
    # 형태소 토크나이즈 결과 문장 리스트
    result_data = list()
    # 반복문을 통해 모든 데이터에 대해 토크나이즈 진행
    for seq in tqdm(data):
        morphlized_seq = " ".join(morph_analyzer.morphs(seq.replace(' ', '')))
        result_data.append(morphlized_seq)

    return result_data
```

```
def enc_processing(value, dictionary):
    sequences_input_index = []
    sequences_length = []

    if DEFINES.tokenize_as_morph:
        value = prepro_like_morphlized(value)


    for sequence in value:
        # FILTERS = "([~.,!?#"':;)(])"
        sequence = re.sub(CHANGE_FILTER, "", sequence)
        sequence_index = []
        for word in sequence.split():
            if dictionary.get(word) is not None:
                sequence_index.extend([dictionary[word]])
            # 잘려진 단어가 딕셔너리에 존재 하지 않는 경우
            else:
                sequence_index.extend([dictionary[UNK]])
        # 문장 제한 길이보다 길어질 경우 뒤에 토큰을 자르고 있다.
        if len(sequence_index) > DEFINES.max_sequence_length:
            sequence_index = sequence_index[:DEFINES.max_sequence_length]
        sequences_length.append(len(sequence_index))
        # 빈 부분에 PAD(0)를 넣어준다.
        sequence_index += (DEFINES.max_sequence_length - len(sequence_index))
        sequence_index.reverse()
        sequences_input_index.append(sequence_index)
    return np.asarray(sequences_input_index), sequences_length
```

2. Model – Seq2seq (data.py)

Make Vocabulary

Index to Word

```python
def pred2string(value, dictionary):
    sentence_string = []
    # 인덱스 배열 하나를 꺼내기.
    for v in value:
            # 딕셔너리에 있는 단어로 변경
            sentence_string = [dictionary[index] for index in v['indexs']]

    print(sentence_string)
    answer = ""
    # 패딩은 모두 스페이스 처리
    for word in sentence_string:
        if word not in PAD and word not in END:
            answer += word
            answer += " "
    print(answer)
    return answer
```

```python
def load_vocabulary():
    vocabulary_list = []
    # 그 파일의 존재 유무를 확인한다.
    if (not (os.path.exists(DEFINES.vocabulary_path))):
        if (os.path.exists(DEFINES.data_path)):
            data_df = pd.read_csv(DEFINES.data_path, encoding='utf-8')
            question, answer = list(data_df['Q']), list(data_df['A'])
            if DEFINES.tokenize_as_morph:  # 형태소에 따른 토크나이져 처리
                question = prepro_like_morphlized(question)
                answer = prepro_like_morphlized(answer)
            data = []
            data.extend(question)
            data.extend(answer)
            words = data_tokenizer(data)
            words = list(set(words))
            # PAD = "<PADDING>"
            # STD = "<START>"
            # END = "<END>"
            # UNK = "<UNKNOWN>"
            words[:0] = MARKER
        # 사전을 리스트로 만들었으니 이 내용을
        # 사전 파일을 만들어 넣는다.
        with open(DEFINES.vocabulary_path, 'w', encoding='utf-8') as vocabulary_file:
            for word in words:
                vocabulary_file.write(word + '\n')
    with open(DEFINES.vocabulary_path, 'r', encoding='utf-8') as vocabulary_file:
        for line in vocabulary_file:
            vocabulary_list.append(line.strip())

    char2idx, idx2char = make_vocabulary(vocabulary_list)
    return char2idx, idx2char, len(char2idx)
```


KU-BIG

2. Model – Seq2seq (model.py)

LSTM Cell

```
def make_lstm_cell(mode, hiddenSize, index):
    cell = tf.nn.rnn_cell.BasicLSTMCell(hiddenSize, name="lstm" + str(index), state_is_tuple=False)
    if mode == tf.estimator.ModeKeys.TRAIN:
        cell = tf.contrib.rnn.DropoutWrapper(cell, state_keep_prob=DEFINES.dropout_width)
    return cell
```

Word Embedding

```
if params['embedding'] == True:
   initializer = tf.contrib.layers.xavier_initializer()
   embedding_encoder = tf.get_variable(name="embedding_encoder",
                                       shape=[params['vocabulary_length'], params['embedding_size']],
                                       dtype=tf.float32,
                                       initializer=initializer,
                                       trainable=True)
else:
    embedding_encoder = tf.eye(num_rows=params['vocabulary_length'], dtype=tf.float32)
    embedding_encoder = tf.get_variable(name="embedding_encoder",
                                       initializer=embedding_encoder,
                                       trainable=False)
```

Encoder & Decoder

```
with tf.variable_scope('encoder_scope', reuse=tf.AUTO_REUSE):
    if params['multilayer'] == True:
        encoder_cell_list = [make_lstm_cell(mode, params['hidden_size'], i) for i in range(params['
        rnn_cell = tf.contrib.rnn.MultiRNNCell(encoder_cell_list, state_is_tuple=False)
    else:
        rnn_cell = make_lstm_cell(mode, params['hidden_size'], "")
    encoder_outputs, encoder_states = tf.nn.dynamic_rnn(cell=rnn_cell,
                                                        inputs=embedding_encoder_batch,
                                                        dtype=tf.float32)
```

2. Model – Seq2seq (model.py)

Calculate Loss & Optimizing

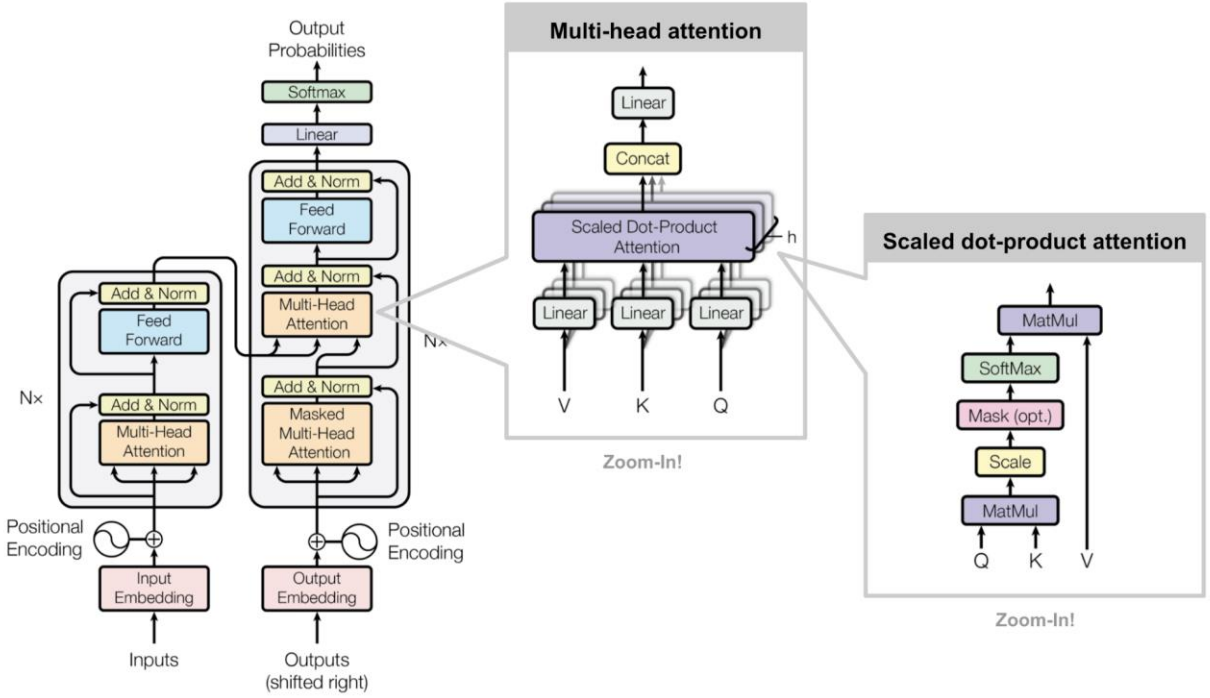
```
if TRAIN and params['loss_mask'] == True:
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=logits, labels=labels_))
    masks = features['length']

    loss = loss * tf.cast(masks, tf.float32)
    loss = tf.reduce_mean(loss)
else:
   loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=logits, labels=labels_))
accuracy = tf.metrics.accuracy(labels=labels, predictions=predict, name='accOp')
```

```
# 아담 옵티마이저 사용
optimizer = tf.train.AdamOptimizer(learning_rate=DEFINES.learning_rate)
# 에러값을 옵티마이저를 사용해서 최소화 시킨다.
train_op = optimizer.minimize(loss, global_step=tf.train.get_global_step())
return tf.estimator.EstimatorSpec(mode, loss=loss, train_op=train_op)
```

2. Model - Transformer

Model
Encoder + Decoder



2. Model - Transformer

HyperParameter

```python
# -*- coding: utf-8 -*-
import tensorflow as tf

tf.app.flags.DEFINE_integer('batch_size', 64, 'batch size')  # 배치 크기
tf.app.flags.DEFINE_integer('train_steps', 20000, 'train steps')  # 학습 에포크
tf.app.flags.DEFINE_float('dropout_width', 0.5, 'dropout width')  # 드롭아웃 크기
tf.app.flags.DEFINE_integer('embedding_size', 128, 'embedding size')  # 가중치 크기 # 논문 512 사용
tf.app.flags.DEFINE_float('learning_rate', 1e-3, 'learning rate')  # 학습률
tf.app.flags.DEFINE_integer('shuffle_seek', 1000, 'shuffle random seek')  # 셔플 시드값
tf.app.flags.DEFINE_integer('max_sequence_length', 25, 'max sequence length')  # 시퀀스 길이
tf.app.flags.DEFINE_integer('model_hidden_size', 128, 'model weights size')  # 모델 가중치 크기
tf.app.flags.DEFINE_integer('ffn_hidden_size', 512, 'ffn weights size')  # ffn 가중치 크기
tf.app.flags.DEFINE_integer('attention_head_size', 4, 'attn head size')  # 멀티 헤드 크기
tf.app.flags.DEFINE_integer('layer_size', 2, 'layer size')  # 논문은 6개 레이어이나 2개 사용 학습 속도 및 성능 튜닝
tf.app.flags.DEFINE_string('data_path', '../data_in/ChatBotData.csv', 'data path')  # 데이터 위치
tf.app.flags.DEFINE_string('vocabulary_path', './data_out/vocabularyData.voc', 'vocabulary path')  # 사전 위치
tf.app.flags.DEFINE_string('check_point_path', './data_out/check_point', 'check point path')  # 체크 포인트 위치
tf.app.flags.DEFINE_boolean('tokenize_as_morph', False, 'set morph tokenize')  # 형태소에 따른 토크나이징 사용 유무
tf.app.flags.DEFINE_boolean('xavier_initializer', True, 'set xavier initializer')  # 형태소에 따른 토크나이징 사용 유무

# Define FLAGS
DEFINES = tf.app.flags.FLAGS
```

2. Model - Transformer

Residual Connection

```python
def layer_norm(inputs, eps=1e-6):
    # LayerNorm(x + Sublayer(x))
    feature_shape = inputs.get_shape()[-1:]
    #  평균과 표준편차을 넘겨 준다.
    mean = tf.keras.backend.mean(inputs, [-1], keepdims=True)
    std = tf.keras.backend.std(inputs, [-1], keepdims=True)
    beta = tf.get_variable("beta", initializer=tf.zeros(feature_shape))
    gamma = tf.get_variable("gamma", initializer=tf.ones(feature_shape))

    return gamma * (inputs - mean) / (std + eps) + beta


def sublayer_connection(inputs, sublayer, dropout=0.2):
    outputs = layer_norm(inputs + tf.keras.layers.Dropout(dropout)(sublayer))
    return outputs
```

Positional Encoding

```python
def positional_encoding(dim, sentence_length):
    encoded_vec = np.array([pos/np.power(10000, 2*i/dim)
                            for pos in range(sentence_length) for i in range(dim)])

    encoded_vec[::2] = np.sin(encoded_vec[::2])
    encoded_vec[1::2] = np.cos(encoded_vec[1::2])

    return tf.constant(encoded_vec.reshape([sentence_length, dim]), dtype=tf.float32)
```

2. Model - Transformer

Multihead Attention

```python
class MultiHeadAttention(tf.keras.Model):
    def __init__(self, num_units, heads, masked=False):
        super(MultiHeadAttention, self).__init__()

        self.heads = heads
        self.masked = masked

        self.query_dense = tf.keras.layers.Dense(num_units, activation=tf.nn.relu)
        self.key_dense = tf.keras.layers.Dense(num_units, activation=tf.nn.relu)
        self.value_dense = tf.keras.layers.Dense(num_units, activation=tf.nn.relu)

    def scaled_dot_product_attention(self, query, key, value, masked=False):
        key_seq_length = float(key.get_shape().as_list()[-1])
        key = tf.transpose(key, perm=[0, 2, 1])
        outputs = tf.matmul(query, key) / tf.sqrt(key_seq_length)

        if masked:
            diag_vals = tf.ones_like(outputs[0, :, :])
            tril = tf.linalg.LinearOperatorLowerTriangular(diag_vals).to_dense()
            masks = tf.tile(tf.expand_dims(tril, 0), [tf.shape(outputs)[0], 1, 1])

            paddings = tf.ones_like(masks) * (-2 ** 32 + 1)
            outputs = tf.where(tf.equal(masks, 0), paddings, outputs)

        attention_map = tf.nn.softmax(outputs)

        return tf.matmul(attention_map, value)
```

2. Model - Transformer

Multihead Attention

```
def call(self, query, key, value):
    query = self.query_dense(query)
    key = self.key_dense(key)
    value = self.value_dense(value)

    query = tf.concat(tf.split(query, self.heads, axis=-1), axis=0)
    key = tf.concat(tf.split(key, self.heads, axis=-1), axis=0)
    value = tf.concat(tf.split(value, self.heads, axis=-1), axis=0)

    attention_map = self.scaled_dot_product_attention(query, key, value, self.masked)

    attn_outputs = tf.concat(tf.split(attention_map, self.heads, axis=0), axis=-1)

    return attn_outputs
```

Feed Forward Network

```
class PositionWiseFeedForward(tf.keras.Model):
    def __init__(self, num_units, feature_shape):
        super(PositionWiseFeedForward, self).__init__()

        self.inner_dense = tf.keras.layers.Dense(num_units, activation=tf.nn.relu)
        self.output_dense = tf.keras.layers.Dense(feature_shape)

    def call(self, inputs):
        inner_layer = self.inner_dense(inputs)
        outputs = self.output_dense(inner_layer)

        return outputs
```

3. Project result

‹Seq2seq›

어디가 – 갈 데가 있어 (ㅇ)
나랑 얘기좀 해 구준표- 할 얘기 없어 다신 아는 척 하지 말랬지 (ㅇ)
구준표 너 진짜 이럴거야?- 여기 너희가 싫어 (x)
잔디는 놔둬- 안 놔두면 어쩔건데 (ㅇ)
누가 너한테 이딴 거 해달랬어? - 야 서민 좋으면 솔직하게 좋다구 말해 댕겨 끝나면 빨랑빨랑 집에 올것이지 (x)
나도 어쩔수 없어 - 어쩌자고 너같은 애를 좋아하게 된걸까 (ㅇ)
죄송해요 제가 연락해볼게요 – 핥아 (x)
부모님이 속상해 하실거야 - 사람이 말을 하는데 어딜 가 (x)
뭐야 왜왔어? - 볼일 있으니까 왔지 (ㅇ)
헉! 이거 뭐야? – 줘봐 (ㅇ)
준표야 너 오랜만에 즐거워보여 - 당연하지 친구들이랑 이렇게 맛있는 저녁을 먹는데 (ㅇ)
그래도 퇴학에 제명은 너무 가혹해 - 지금쯤 뚜껑 제대로 열렸을거다 (ㅇ)
누구 기다려? - 사람이 말을 하는데 어딜 가 (x)
할 얘기 있어? - 난 없어 (ㅇ)
맛없겠지? - 너 말을 하는데 어딜 가 (x)
내가 지킬거야 - 넌 대체 무식한거냐 요령이 없는거냐 (ㅇ)
왜 이렇게 뜨거워 너 괜찮아? - 나 아파 (ㅇ)
너 어제 진짜 금잔디네 갔어? - 니들 김장해본 적 있어 (ㅇ)
아직도 못 믿는구나?- 믿는다구 젠장 (ㅇ)
우와 별이다 - 바보 아냐 (ㅇ)

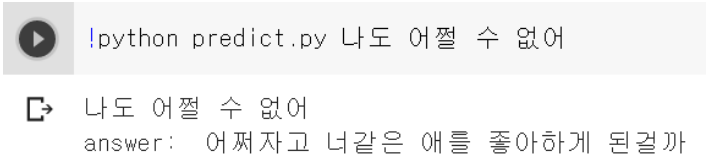
‹Transformer›

1. 어디가-갈 데가 있어
2. 나랑 얘기좀 해 구준표-할 얘기 없어 다신 아는 척 하지 말랬지
3. 구준표 너 진짜 이럴거야?-니들 김장해본 적 있어
4. 잔디는 놔둬-안 놔두면 어쩔건데
5. 누가 너한테 이딴거 해 달랬어?-야 서민 좋으면 솔직하게 좋다고 말해
6. 나도 어쩔 수 없어-어쩌자고 너같은 애를 좋아하게 된걸까
7. 죄송해요 제가 연락해볼게요-다들 저리 안가
8. 부모님이 속상해 하실거야-그러니까 앞으로 금잔디한테 누구든 까불지 마라
9. 뭐야 왜왔어?-볼일 있으니까 왔지
10. 헉! 이거 뭐야?-줘봐
11. 준표야 너 오랜만에 즐거워보여-당연하지 친구들이랑 이렇게 맛있는 저녁을 먹는데
12. 그래도 퇴학에 제명은 너무 가혹해-그 딴 걸 어떻게 고쳐서 입어
13. 누구 기다려?-너희 동네선 사과를 이 따위로 하냐
14. 난 할 얘기 있어-난 없어
15. 맛 없겠지?-한 번 줬으면 내꺼지 어딜 건드려
16. 내가 지킬거야-다들 죽고 싶어
17. 왜 이렇게 뜨거워 너 괜찮아?-나 아파
18. 너 어제 진짜 금잔디네 갔어?-니들 김장해본 적 있어?
19. 아직도 못 믿는구나?-믿는다구 젠장
20. 우와 별이다-바보 아냐

KU-BIG

4. Need to Complement

1. 데이터의 품질
   - 383개의 data
   - 드라마 대본



2. 단순했던 구조
   - embedding layer => xavier
   - encoder, decoder

5. CHATBOT

나랑 얘기하고 싶어?

KU-BIG