Mission 2

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import konlpy as kp
```

In [2]:
```
df1 = pd.read_csv('배달의_민족_기업평가.csv')
```

In [6]:
```
# df1 데이터를 이용해 아래의 미션을 수행하고자 한다
# 1) '작성시간'데이터를 '작성시간_dt'변수로 날짜데이터로 변환하여 선언
df1['작성시간_dt'] = pd.to_datetime(df1['작성시간'])
```

In [10]:
```
# 2) '재직여부'에서 '전직원'들이 가장 많이 리뷰를 남긴 연도를 확인
df1['작성연도'] = df1['작성시간_dt'].dt.year

cond1 = (df1['재직여부']=='전직원')
df1.loc[cond1]['작성연도'].value_counts()
```

Out[10]:
```
2019    32
2018    27
2017    15
2016     8
2020     5
2015     3
Name: 작성연도, dtype: int64
```

In [23]:
```
# 3) '직종'중 가장 빈도수가 높은 2 직종을 추출하여,
#    df1_top으로 선언한 뒤, df1_top 에서 '한줄평'에 대한 '명사'를 워드클라우드로 시각
#    -> image.png 파일로 저장
work_order_list = df1['직종'].value_counts().index.tolist()

cond1   = df1['직종'].isin(work_order_list[:2])
df1_top = df1.loc[cond1]
```

In [24]:
```
# 시리즈 -> 형태소 분석
def pos_dataframe(data):
    okt = kp.tag.Okt()
    df_POS = pd.DataFrame()
    for i in range(0, len(data)):
        dfn = pd.DataFrame( okt.pos(data.values.tolist()[i]) )
        df_POS = pd.concat([df_POS, dfn])
    return df_POS.rename(columns={0:'형태소',1:'품사'})
```

In [27]:
```
df_Pos = pos_dataframe(df1_top['한줄평'])
cond1 = (df_Pos['품사']=='Noun')
df_N  = df_Pos.loc[cond1]
```

In [30]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [34]:
```
wc_img = WordCloud(background_color='white', width=800, height=800,
                   font_path='Malgun.ttf').generate(' '.join(df_N['형태소']))
plt.figure(figsize=[10,10])
plt.imshow(wc_img)
plt.show()
plt.savefig('img_wc.png')
```

<Figure size 432x288 with 0 Axes>

In [39]:
```
# - company_Train_data.csv 데이터를 불러와 아래의 미션을 수행하고자 한다.
# 1) 해당 데이터를 df3로 불러와, 한줄평에 따른 '기업성장여부'를 분류하는 분류모델을
df3 = pd.read_csv('company_Train_data.csv')
```

In [43]:
```
Y = df3['기업성장여부'].replace({'성장':1, '정체':0})
X = df3[['한줄평']]
```

In [44]:
```
from sklearn.model_selection import train_test_split
```

In [45]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state=1234)
```

In [48]:
```
# 불용어 처리함수
import re
def text_remove_stopword(i):
    review_text = re.sub('[!1+(),.@#2345]?\n', "", i)
    word_text = kp.tag.Okt().morphs(review_text, stem=True)
    return word_text
```

In [49]:
```
X_train_list = []
X_test_list  = []
```

In [51]:
```
for i in X_train['한줄평']:
    if type(i)==str:
        X_train_list.append(text_remove_stopword(i))
```

```
    else:
        X_train_list.append( [] )

for i in X_test['한줄평']:
    if type(i)==str:
        X_test_list.append(text_remove_stopword(i))
    else:
        X_test_list.append( [] )
```

In [52]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [55]:
```
# Text to Seq / Padding
tts_Vec = Tokenizer()
tts_Vec.fit_on_texts(X_train_list)

train_Seq = tts_Vec.texts_to_sequences(X_train_list)
test_Seq  = tts_Vec.texts_to_sequences(X_test_list)
```

In [56]:
```
train_input = pad_sequences(train_Seq, padding='post', maxlen=50)
test_input  = pad_sequences(test_Seq, padding='post', maxlen=50)
```

In [57]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
```

In [58]:
```
model_pipe = make_pipeline(RobustScaler(), GradientBoostingClassifier())
hyper_parameter = {}
grid_model = GridSearchCV(model_pipe, param_grid=hyper_parameter, cv=5, n_jobs=-1)
grid_model.fit(train_input, Y_train)
```

Out[58]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('robustscaler', RobustScaler()),
                                       ('gradientboostingclassifier',
                                        GradientBoostingClassifier())]),
             n_jobs=-1, param_grid={})
```

In [59]:
```
best_model = grid_model.best_estimator_
```

In [62]:
```
import pickle
pickle.dump(best_model, open('model_nlp.sav','wb'))
```

In [71]:
```
# 3) company_Test_data.csv를 불러와 df3_test로 선언한뒤, 앞서 만든 모델에 넣어
#    '기업성장여부'를 분류하고, 분류된 '기업성장여부' 빈도수 확인
df3_test = pd.read_csv('company_Test_data.csv')
```

In [145…
```
def input_nlp_preprocess(data):
    X_new_list = []
    for i in data['한줄평']:
        if type(i)==str:
            X_new_list.append( text_remove_stopword(i) )
        else:
            X_new_list.append( [] )

    new_seq   = tts_Vec.texts_to_sequences(X_new_list)
    new_input = pad_sequences(new_seq, padding='post', maxlen= 50)
    return new_input
```

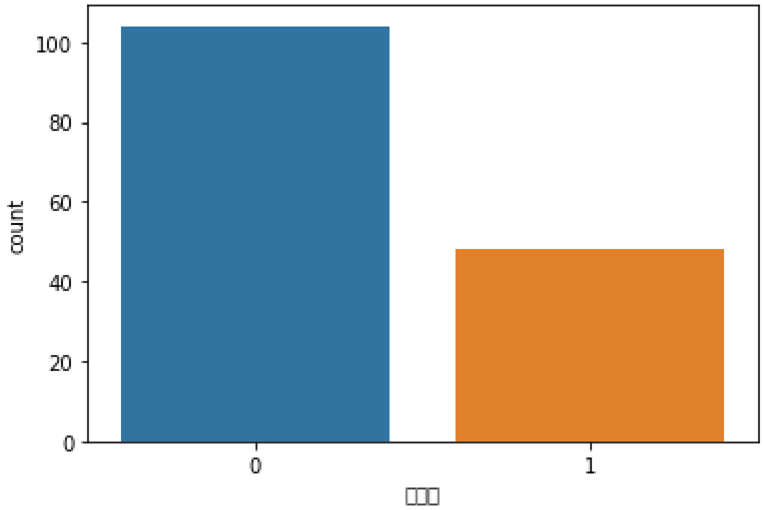
In [146…
```
new_input = input_nlp_preprocess(df3_test[['한줄평']])
```

In [76]:
```
df3_test['분류값'] = best_model.predict(new_input)
```

In [77]:
```
sns.countplot(data=df3_test, x='분류값')
```

Out[77]:
```
<AxesSubplot:xlabel='분류값', ylabel='count'>
```

```
C:₩Users₩DMC CONET₩AppData₩Roaming₩jupyterlab-desktop₩jlab_server₩lib₩site-packages
₩IPython₩core₩pylabtools.py:151: UserWarning: Glyph 48516 (₩N{HANGUL SYLLABLE BUN})
missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:₩Users₩DMC CONET₩AppData₩Roaming₩jupyterlab-desktop₩jlab_server₩lib₩site-packages
₩IPython₩core₩pylabtools.py:151: UserWarning: Glyph 47448 (₩N{HANGUL SYLLABLE RYU})
missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:₩Users₩DMC CONET₩AppData₩Roaming₩jupyterlab-desktop₩jlab_server₩lib₩site-packages
₩IPython₩core₩pylabtools.py:151: UserWarning: Glyph 44050 (₩N{HANGUL SYLLABLE GABS})
missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [78]:
```
df3_test['분류값'].value_counts()
```

Out[78]:
```
0    104
1     48
Name: 분류값, dtype: int64
```

텍스트 유사도

- 서로 비슷한 텍스트가 얼마나 유사한지를 표현
- 문장의 구조는 다르지만 의미가 같은 경우 / 동의어를 사용해 문장이 구성된 경우
- 네트워크 분석 : 단어간의 유사성 (연관분석 / 특정 단어가 같이 등장할 확률) /시각화
- 신경망 분석 : 판별모델에서 Encoder 구성 / 생성모델에서 Transfomer 구성

In [81]:
```
# networkx : 네트워크 분석 시각화 라이브러리 / apyori : 연관분석
# !pip install --user networkx
# !pip install --user apyori
```

In [83]:
```
# 네트워크 분석
# 형태소 분석 -> 유사도 계산 -> 네트워크 시각화
df3['한줄평']
```

Out[83]:
```
0      1. 내부구성원들을 위하는 기업문화가 형성되어있고, 실제로 운영측면에서도 배려마인드...
1      그 누구의 눈치도 보지 않으며, 자유로운 의사소통에 내부 분위기, 팀별로 케바케이...
2                 도서구입비 무제한 지원, 주 35시간 근무, 월요일은 오후1시 출근,
3      자율적인 분위기와 좋은 복지를 많이 갖춘 곳. 좋은 사람들이 많이 모여있고 그 안에...
4      1. 내부구성원들을 위하는 기업문화가 형성되어있고, 실제로 운영측면에서도 배려마인드...
                             ...
299    딱히 없음. 회사가 좋아서 여기 있다가 이직하실때 오히려 다른 회사에 적응못할수있...
300    대기업처럼 체계적으로 시스템이 많이 갖추어지지는 않았어요. 그래서 지금 열심히만들...
301                   로테이션 근무₩n교육 커리큘럼의 체계가 잡혀있지 않음
302    아이티는 어쩔수 없다. 업무량대폭발. !!!! 그래도 야근은 종종 철야는 아주아주...
303                      편하게 놀거라고 생각하고 오면 당황할 수 있다.
Name: 한줄평, Length: 304, dtype: object
```

In [88]:
```
# 불용어 파일 불러와 처리
df_stopword  = pd.read_csv('stopword.txt', header=None, names=['불용어'])
stopword_set = set(df_stopword['불용어'].values.tolist())
```

In [94]:
```
# 불용어 처리함수
def text_remove_stopword(i):
    review_text = re.sub('[!1+(),.@#2345]?₩n', "", i)
    word_text = kp.tag.Okt().morphs(review_text, stem=True)
    word_text2 = [ tk for tk in word_text if not tk in stopword_set ]
    return word_text2
```

In [103...
```
clean_word = []
for i in df3['한줄평']:
    if type(i)==str:
        clean_word.append( text_remove_stopword(i) )
    else :
        clean_word.append( [] )
```

In [104...
```
import apyori
```

In [139...
```
# 지지도 Support : 전체 데이터에서 A와 B 가 동시에 포함된 비율
result_apy  = apyori.apriori(clean_word, min_support = 0.03)
df_support  = pd.DataFrame(list(result_apy))
```

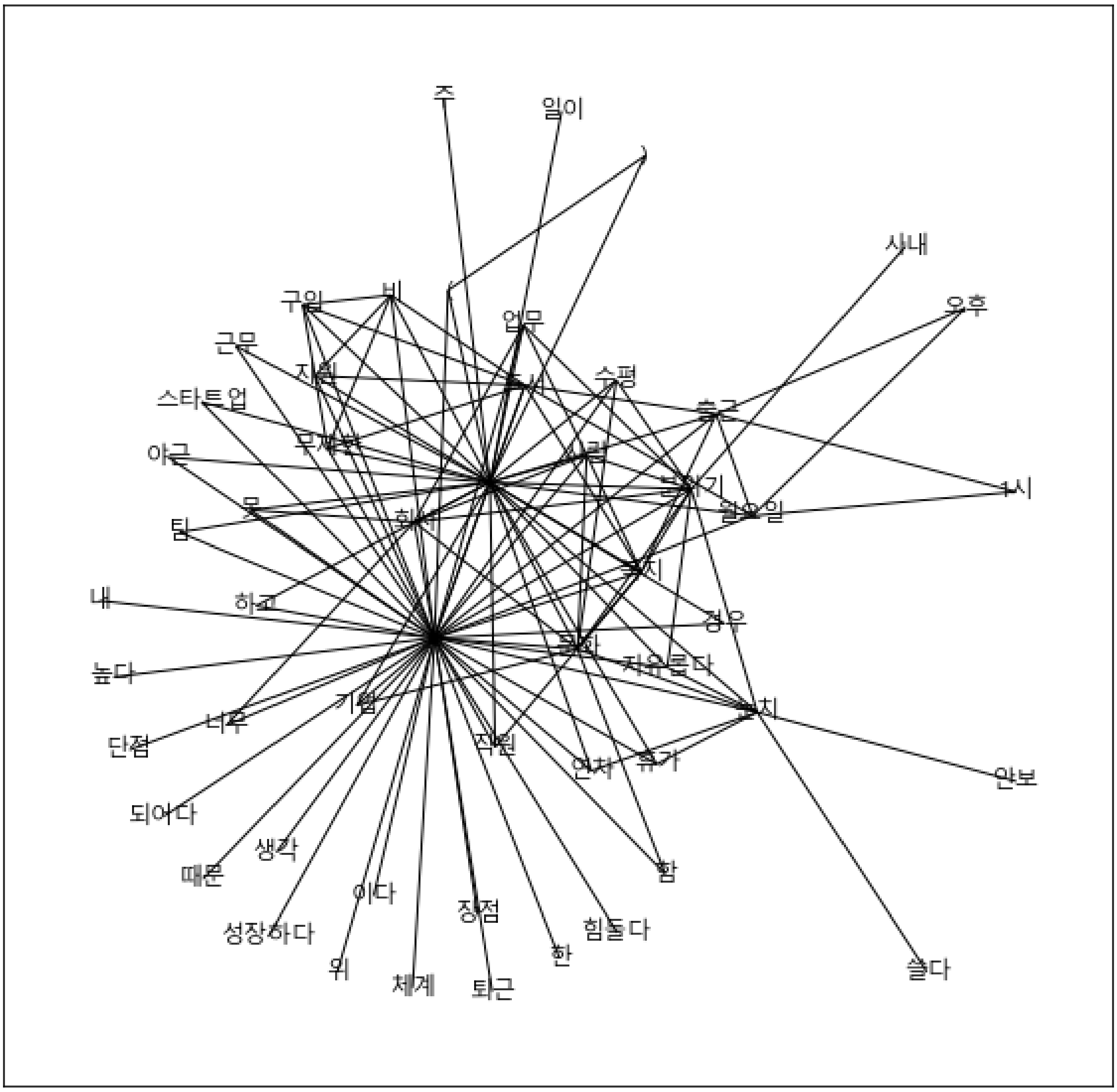
In [140...
```
df_support['count'] = df_support['items'].apply( lambda x : len(x))
```

In [141...
```
cond1 = (df_support['count']==2)
df_network = df_support.loc[cond1]
```

In [142...
```
import networkx as nx
```

In [143...
```
graph_model = nx.Graph()
pr          = nx.pagerank(graph_model)
graph_model.add_edges_from(df_network['items'])
```

In [144...
```
grp = nx.kamada_kawai_layout(graph_model)
plt.figure(figsize=[10,10])
nx.draw_networkx(graph_model, font_family='Malgun Gothic',
                 pos = grp,
                 node_color=list(pr.values()))
```



신경망 알고리즘 활용해 분류모델 구성

In [153…

```
!pip install --user keras
```

```
Requirement already satisfied: keras in c:\users\dmc conet\appdata\roaming\jupyterlab-desktop\jlab_server\lib\site-packages (2.11.0)
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\dmc conet\appdata\roaming\jupyterlab-desktop\jlab_server\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\dmc conet\appdata\roaming\jupyterlab-desktop\jlab_server\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\dmc conet\appdata\roaming\jupyterlab-desktop\jlab_server\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\dmc conet\appdata\roaming\jupyterlab-desktop\jlab_server\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\dmc conet\appdata\roaming\jupyterlab-desktop\jlab_server\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\dmc conet\appdata\roaming\jupyterlab-desktop\jlab_server\lib\site-packages)
```

In [154…

```
import tensorflow
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop, SGD
from keras.utils import to_categorical
```

In [157…

```
# Y 값에 대한 Matrix 구성
```

```
Y_matrix = to_categorical(Y_train)
```

In [166…

```
# Model 구성
model1 = Sequential()
model1.add(layers.Embedding(1600, 32, input_length = 50 ))
model1.add(layers.SimpleRNN(32)),
model1.add(layers.Dense(2, activation='sigmoid'))
model1.summary()

model1.compile(optimizer=RMSprop(), loss='binary_crossentropy', metrics= ['acc'])

history1 = model1.fit(train_input, Y_matrix, epochs=40, batch_size=32,
                      validation_split = 0.2)
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 50, 32)            51200

 simple_rnn_4 (SimpleRNN)    (None, 32)                2080

 dense_4 (Dense)             (None, 2)                 66

=================================================================
Total params: 53,346
Trainable params: 53,346
Non-trainable params: 0
_________________________________________________________________
Epoch 1/40
6/6 [==============================] - 2s 251ms/step - loss: 0.6991 - acc: 0.5165 -
val_loss: 0.7064 - val_acc: 0.4565
Epoch 2/40
6/6 [==============================] - 0s 13ms/step - loss: 0.6021 - acc: 0.8297 - v
al_loss: 0.6440 - val_acc: 0.6522
Epoch 3/40
6/6 [==============================] - 0s 13ms/step - loss: 0.4917 - acc: 0.9451 - v
al_loss: 0.6691 - val_acc: 0.6304
Epoch 4/40
6/6 [==============================] - 0s 12ms/step - loss: 0.3903 - acc: 0.9835 - v
al_loss: 0.6451 - val_acc: 0.6522
Epoch 5/40
6/6 [==============================] - 0s 12ms/step - loss: 0.3299 - acc: 0.9835 - v
al_loss: 0.6027 - val_acc: 0.7174
Epoch 6/40
6/6 [==============================] - 0s 12ms/step - loss: 0.2825 - acc: 0.9945 - v
al_loss: 0.6020 - val_acc: 0.6957
Epoch 7/40
6/6 [==============================] - 0s 12ms/step - loss: 0.2133 - acc: 0.9945 - v
al_loss: 0.5673 - val_acc: 0.7826
Epoch 8/40
6/6 [==============================] - 0s 12ms/step - loss: 0.1853 - acc: 0.9945 - v
al_loss: 0.6072 - val_acc: 0.6739
Epoch 9/40
6/6 [==============================] - 0s 12ms/step - loss: 0.1569 - acc: 1.0000 - v
al_loss: 0.5627 - val_acc: 0.7174
Epoch 10/40
6/6 [==============================] - 0s 13ms/step - loss: 0.1143 - acc: 1.0000 - v
al_loss: 0.5704 - val_acc: 0.6739
Epoch 11/40
6/6 [==============================] - 0s 12ms/step - loss: 0.1067 - acc: 1.0000 - v
al_loss: 0.5723 - val_acc: 0.6739
Epoch 12/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0871 - acc: 1.0000 - v
al_loss: 0.5477 - val_acc: 0.7391
Epoch 13/40
6/6 [==============================] - 0s 11ms/step - loss: 0.0625 - acc: 1.0000 - v
al_loss: 0.5582 - val_acc: 0.7609
Epoch 14/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0507 - acc: 1.0000 - v
al_loss: 0.5570 - val_acc: 0.7609
Epoch 15/40
6/6 [==============================] - 0s 13ms/step - loss: 0.0430 - acc: 1.0000 - v
al_loss: 0.6040 - val_acc: 0.6522
Epoch 16/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0591 - acc: 1.0000 - v
al_loss: 0.5849 - val_acc: 0.6957
Epoch 17/40
```

```
6/6 [==============================] - 0s 12ms/step - loss: 0.0338 - acc: 1.0000 - v
al_loss: 0.5629 - val_acc: 0.7391
Epoch 18/40
6/6 [==============================] - 0s 11ms/step - loss: 0.0259 - acc: 1.0000 - v
al_loss: 0.5563 - val_acc: 0.7391
Epoch 19/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0215 - acc: 1.0000 - v
al_loss: 0.5628 - val_acc: 0.7609
Epoch 20/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0181 - acc: 1.0000 - v
al_loss: 0.5608 - val_acc: 0.7391
Epoch 21/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0186 - acc: 1.0000 - v
al_loss: 0.5909 - val_acc: 0.7391
Epoch 22/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0296 - acc: 1.0000 - v
al_loss: 0.5646 - val_acc: 0.7174
Epoch 23/40
6/6 [==============================] - 0s 11ms/step - loss: 0.0138 - acc: 1.0000 - v
al_loss: 0.5983 - val_acc: 0.7174
Epoch 24/40
6/6 [==============================] - 0s 11ms/step - loss: 0.0103 - acc: 1.0000 - v
al_loss: 0.6080 - val_acc: 0.7174
Epoch 25/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0088 - acc: 1.0000 - v
al_loss: 0.6192 - val_acc: 0.7174
Epoch 26/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0075 - acc: 1.0000 - v
al_loss: 0.6207 - val_acc: 0.7174
Epoch 27/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0064 - acc: 1.0000 - v
al_loss: 0.6299 - val_acc: 0.7174
Epoch 28/40
6/6 [==============================] - 0s 11ms/step - loss: 0.0055 - acc: 1.0000 - v
al_loss: 0.6340 - val_acc: 0.7174
Epoch 29/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0046 - acc: 1.0000 - v
al_loss: 0.6382 - val_acc: 0.6957
Epoch 30/40
6/6 [==============================] - 0s 11ms/step - loss: 0.0261 - acc: 1.0000 - v
al_loss: 0.6851 - val_acc: 0.7174
Epoch 31/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0051 - acc: 1.0000 - v
al_loss: 0.6866 - val_acc: 0.7174
Epoch 32/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0037 - acc: 1.0000 - v
al_loss: 0.6776 - val_acc: 0.7391
Epoch 33/40
6/6 [==============================] - 0s 11ms/step - loss: 0.0033 - acc: 1.0000 - v
al_loss: 0.6580 - val_acc: 0.7391
Epoch 34/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0030 - acc: 1.0000 - v
al_loss: 0.6444 - val_acc: 0.7391
Epoch 35/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0027 - acc: 1.0000 - v
al_loss: 0.6290 - val_acc: 0.7609
Epoch 36/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0025 - acc: 1.0000 - v
al_loss: 0.6149 - val_acc: 0.7609
Epoch 37/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0022 - acc: 1.0000 - v
al_loss: 0.5975 - val_acc: 0.7391
Epoch 38/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0020 - acc: 1.0000 - v
```

```
al_loss: 0.5881 - val_acc: 0.7391
Epoch 39/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0017 - acc: 1.0000 - v
al_loss: 0.5814 - val_acc: 0.7826
Epoch 40/40
6/6 [==============================] - 0s 12ms/step - loss: 0.0015 - acc: 1.0000 - v
al_loss: 0.5706 - val_acc: 0.7391
```

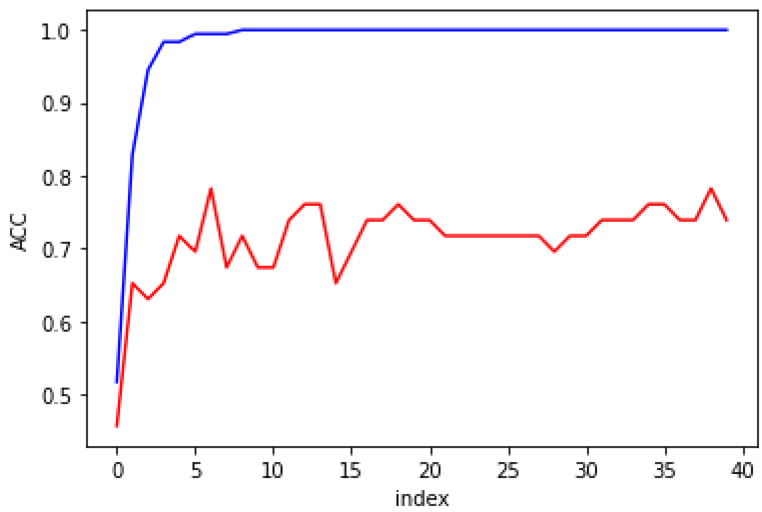
In [171…

```
df_score = pd.DataFrame()
df_score['ACC'] = history1.history['acc']
df_score['Loss'] = history1.history['loss']
df_score['val_ACC'] = history1.history['val_acc']
df_score['val_Loss'] = history1.history['val_loss']
df_score2 = df_score.reset_index()
```

In [174…

```
sns.lineplot(data=df_score2, x='index', y='ACC', color='b')
sns.lineplot(data=df_score2, x='index', y='val_ACC' , color='r')
```

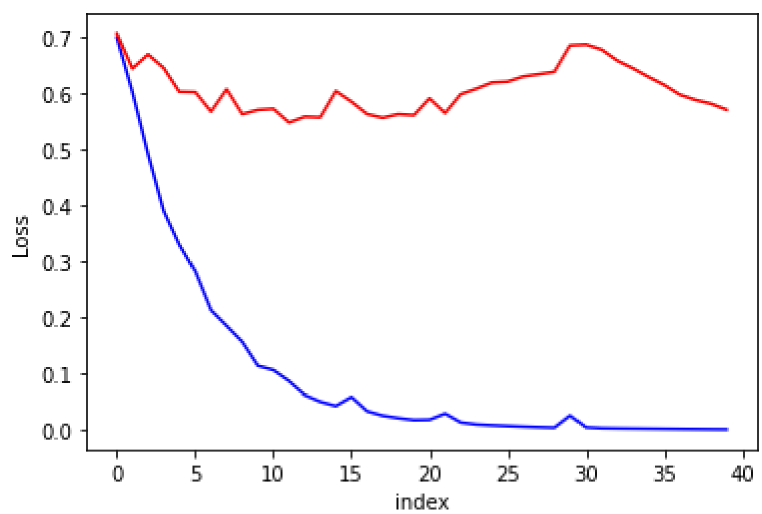
Out[174]: <AxesSubplot:xlabel='index', ylabel='ACC'>



In [175…

```
sns.lineplot(data=df_score2, x='index', y='Loss', color='b')
sns.lineplot(data=df_score2, x='index', y='val_Loss' , color='r')
```

Out[175]: <AxesSubplot:xlabel='index', ylabel='Loss'>



1. **RNN**

- Recurrent Neural Network (RNN): 순서를 갖는 (Sequence) 데이터를 학습 / 처리

- Node (Unit)에서 처리된 정보를 'State' 로 저장하여 앞/뒤 정보를 유지하며 학습 (순환연결)
- 한계 : 긴 문장(Sequence)이 오게되면 이전에 처리되었던 state를 적절하게 유지하기 어려움
  - Vanishing Gradient (기울기 소실) : Layer가 많은 신경망 알고리즘에서 주로 발생
  - 신경망 앞쪽에 있는 Layer의 Weight가 적절하게 Update 되지 못함
  - 오차(Loss)가 더 이상 줄어들지 않고 특정 값에 Weight값이 고정

1. **LSTM (Long Short Term Memory)**

- Vanishing Gradient (기울기 소실)의 문제를 개선하기 위해, 앞선 Node(Unit)처리된 정보를 지속적으로 유지할 수 있는 한의 **State Node**를 구성하여 학습
  - 순환 드롭 아웃 : Overfitting 방지
  - 스태킹 Layer : Layer 연산 강화 (Cost 비용)
  - 양방향 순환 Layer : 같은 정보를 다른 방향으로 주입하여 학습

1. **GRU(Gated Recurrent Unit)**
   - LSTM 불필요한 구조를 제거 한 형태의 신경망 알고리즘
   - 순환 드롭 아웃 : Layer 내 연산을 수행하는 Node(Unit)를 랜덤으로 건너뛰어 연산
     - LSTM 연산의 량은 줄이고, 과적합도 방지 할 수 있지만, 성능이 개선되지는 않음
     - 모든 Unit 내에 동일한 Drop Out 적용

LSTM

In [206…

```
model2 = Sequential() # Layer를 구성하기위한 공간을 객체로 선언
model2.add(layers.Embedding(1600, 64, input_length= 50 ))
model2.add(layers.LSTM(32))
model2.add(layers.Dense(2, activation='sigmoid'))
model2.compile(optimizer=RMSprop(), loss='binary_crossentropy', metrics=['acc'])
model2.summary()

history = model2.fit(train_input, Y_matrix, epochs=40, batch_size=32,
                     validation_split = 0.2)
```

```
Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_18 (Embedding)    (None, 50, 64)            102400

 lstm_6 (LSTM)               (None, 32)                12416

 dense_15 (Dense)            (None, 2)                 66

=================================================================
Total params: 114,882
Trainable params: 114,882
Non-trainable params: 0
_________________________________________________________________
Epoch 1/40
6/6 [==============================] - 3s 196ms/step - loss: 0.6942 - acc: 0.5000 - val_loss: 0.6980 - val_acc: 0.4348
Epoch 2/40
6/6 [==============================] - 0s 19ms/step - loss: 0.6912 - acc: 0.5220 - val_loss: 0.6995 - val_acc: 0.4348
Epoch 3/40
6/6 [==============================] - 0s 19ms/step - loss: 0.6896 - acc: 0.5495 - val_loss: 0.6986 - val_acc: 0.4783
Epoch 4/40
6/6 [==============================] - 0s 18ms/step - loss: 0.6874 - acc: 0.5989 - val_loss: 0.6975 - val_acc: 0.4783
Epoch 5/40
6/6 [==============================] - 0s 20ms/step - loss: 0.6747 - acc: 0.6758 - val_loss: 0.6728 - val_acc: 0.5435
Epoch 6/40
6/6 [==============================] - 0s 20ms/step - loss: 0.4892 - acc: 0.8791 - val_loss: 0.4169 - val_acc: 0.8478
Epoch 7/40
6/6 [==============================] - 0s 19ms/step - loss: 0.3119 - acc: 0.9505 - val_loss: 0.3789 - val_acc: 0.9130
Epoch 8/40
6/6 [==============================] - 0s 18ms/step - loss: 0.2640 - acc: 0.9560 - val_loss: 0.4068 - val_acc: 0.8478
Epoch 9/40
6/6 [==============================] - 0s 19ms/step - loss: 0.1916 - acc: 0.9725 - val_loss: 0.3850 - val_acc: 0.8696
Epoch 10/40
6/6 [==============================] - 0s 18ms/step - loss: 0.1424 - acc: 0.9890 - val_loss: 0.3924 - val_acc: 0.8696
Epoch 11/40
6/6 [==============================] - 0s 19ms/step - loss: 0.1202 - acc: 0.9890 - val_loss: 0.4064 - val_acc: 0.8696
Epoch 12/40
6/6 [==============================] - 0s 19ms/step - loss: 0.1052 - acc: 0.9890 - val_loss: 0.4227 - val_acc: 0.8696
Epoch 13/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0939 - acc: 0.9890 - val_loss: 0.4319 - val_acc: 0.8696
Epoch 14/40
6/6 [==============================] - 0s 20ms/step - loss: 0.0696 - acc: 0.9945 - val_loss: 0.4564 - val_acc: 0.8478
Epoch 15/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0617 - acc: 0.9945 - val_loss: 0.4625 - val_acc: 0.8478
Epoch 16/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0556 - acc: 0.9945 - val_loss: 0.4677 - val_acc: 0.8696
Epoch 17/40
```

```
6/6 [==============================] - 0s 19ms/step - loss: 0.0318 - acc: 1.0000 - v
al_loss: 0.4907 - val_acc: 0.8696
Epoch 18/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0267 - acc: 1.0000 - v
al_loss: 0.5128 - val_acc: 0.8696
Epoch 19/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0224 - acc: 1.0000 - v
al_loss: 0.5326 - val_acc: 0.8696
Epoch 20/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0189 - acc: 1.0000 - v
al_loss: 0.5515 - val_acc: 0.8696
Epoch 21/40
6/6 [==============================] - 1s 120ms/step - loss: 0.0159 - acc: 1.0000 -
val_loss: 0.5726 - val_acc: 0.8696
Epoch 22/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0133 - acc: 1.0000 - v
al_loss: 0.5936 - val_acc: 0.8696
Epoch 23/40
6/6 [==============================] - 0s 18ms/step - loss: 0.0112 - acc: 1.0000 - v
al_loss: 0.6134 - val_acc: 0.8696
Epoch 24/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0094 - acc: 1.0000 - v
al_loss: 0.6348 - val_acc: 0.8696
Epoch 25/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0079 - acc: 1.0000 - v
al_loss: 0.6572 - val_acc: 0.8696
Epoch 26/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0066 - acc: 1.0000 - v
al_loss: 0.6789 - val_acc: 0.8696
Epoch 27/40
6/6 [==============================] - 0s 18ms/step - loss: 0.0056 - acc: 1.0000 - v
al_loss: 0.7021 - val_acc: 0.8696
Epoch 28/40
6/6 [==============================] - 0s 18ms/step - loss: 0.0046 - acc: 1.0000 - v
al_loss: 0.7268 - val_acc: 0.8696
Epoch 29/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0039 - acc: 1.0000 - v
al_loss: 0.7503 - val_acc: 0.8696
Epoch 30/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0032 - acc: 1.0000 - v
al_loss: 0.7739 - val_acc: 0.8696
Epoch 31/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0027 - acc: 1.0000 - v
al_loss: 0.7967 - val_acc: 0.8696
Epoch 32/40
6/6 [==============================] - 0s 19ms/step - loss: 0.1988 - acc: 0.9670 - v
al_loss: 1.0172 - val_acc: 0.8261
Epoch 33/40
6/6 [==============================] - 0s 20ms/step - loss: 0.1595 - acc: 0.9725 - v
al_loss: 0.9911 - val_acc: 0.8261
Epoch 34/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0341 - acc: 0.9945 - v
al_loss: 0.8704 - val_acc: 0.8478
Epoch 35/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0036 - acc: 1.0000 - v
al_loss: 0.8765 - val_acc: 0.8478
Epoch 36/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0034 - acc: 1.0000 - v
al_loss: 0.8843 - val_acc: 0.8478
Epoch 37/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0032 - acc: 1.0000 - v
al_loss: 0.8941 - val_acc: 0.8478
Epoch 38/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0030 - acc: 1.0000 - v
```

```
al_loss: 0.9061 - val_acc: 0.8478
Epoch 39/40
6/6 [==============================] - 0s 19ms/step - loss: 0.0027 - acc: 1.0000 - v
al_loss: 0.9203 - val_acc: 0.8478
Epoch 40/40
6/6 [==============================] - 0s 18ms/step - loss: 0.0025 - acc: 1.0000 - v
al_loss: 0.9368 - val_acc: 0.8478
```

In [193…

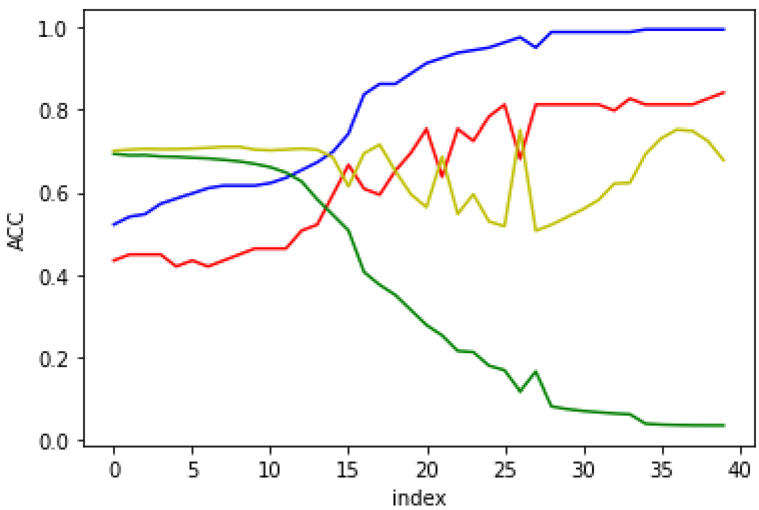
```
def score_df(model):
    df_score = pd.DataFrame()
    df_score['ACC'] = history.history['acc']
    df_score['Loss'] = history.history['loss']
    df_score['val_ACC'] = history.history['val_acc']
    df_score['val_Loss'] = history.history['val_loss']
    df_score2 = df_score.reset_index()
    return df_score2
```

In [194…

```
def eval_plot(df_score2):
    sns.lineplot(data=df_score2, x='index', y='ACC', color='b')
    sns.lineplot(data=df_score2, x='index', y='val_ACC' , color='r')
    sns.lineplot(data=df_score2, x='index', y='Loss', color='g')
    sns.lineplot(data=df_score2, x='index', y='val_Loss' , color='y')
```

In [195…

```
df_lstm_score = score_df(history)
eval_plot(df_lstm_score)
```



GRU

In [196…

```
model3 = Sequential() # Layer를 구성하기위한 공간을 객체로 선언
model3.add(layers.Embedding(1600, 64, input_length= 50 ))
model3.add(layers.GRU(32))
model3.add(layers.Dense(2, activation='sigmoid'))
model3.compile(optimizer=RMSprop(), loss='binary_crossentropy', metrics=['acc'])
model3.summary()

history = model3.fit(train_input, Y_matrix, epochs=40, batch_size=32,
                     validation_split = 0.3)
```

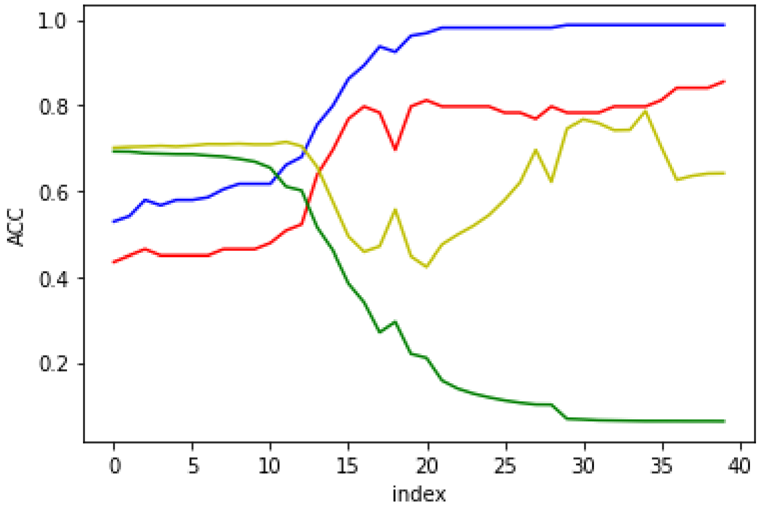
```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, 50, 64)            102400

 gru_2 (GRU)                 (None, 32)                9408

 dense_11 (Dense)            (None, 2)                 66

=================================================================
Total params: 111,874
Trainable params: 111,874
Non-trainable params: 0
_________________________________________________________________
Epoch 1/40
5/5 [==============================] - 2s 113ms/step - loss: 0.6921 - acc: 0.5283 - val_loss: 0.7007 - val_acc: 0.4348
Epoch 2/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6915 - acc: 0.5409 - val_loss: 0.7027 - val_acc: 0.4493
Epoch 3/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6879 - acc: 0.5786 - val_loss: 0.7035 - val_acc: 0.4638
Epoch 4/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6868 - acc: 0.5660 - val_loss: 0.7052 - val_acc: 0.4493
Epoch 5/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6857 - acc: 0.5786 - val_loss: 0.7035 - val_acc: 0.4493
Epoch 6/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6854 - acc: 0.5786 - val_loss: 0.7058 - val_acc: 0.4493
Epoch 7/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6823 - acc: 0.5849 - val_loss: 0.7092 - val_acc: 0.4493
Epoch 8/40
5/5 [==============================] - 0s 20ms/step - loss: 0.6798 - acc: 0.6038 - val_loss: 0.7088 - val_acc: 0.4638
Epoch 9/40
5/5 [==============================] - 0s 20ms/step - loss: 0.6746 - acc: 0.6164 - val_loss: 0.7101 - val_acc: 0.4638
Epoch 10/40
5/5 [==============================] - 0s 20ms/step - loss: 0.6682 - acc: 0.6164 - val_loss: 0.7083 - val_acc: 0.4638
Epoch 11/40
5/5 [==============================] - 0s 20ms/step - loss: 0.6537 - acc: 0.6164 - val_loss: 0.7085 - val_acc: 0.4783
Epoch 12/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6099 - acc: 0.6604 - val_loss: 0.7143 - val_acc: 0.5072
Epoch 13/40
5/5 [==============================] - 0s 21ms/step - loss: 0.6007 - acc: 0.6792 - val_loss: 0.7043 - val_acc: 0.5217
Epoch 14/40
5/5 [==============================] - 0s 20ms/step - loss: 0.5146 - acc: 0.7547 - val_loss: 0.6556 - val_acc: 0.6377
Epoch 15/40
5/5 [==============================] - 0s 20ms/step - loss: 0.4615 - acc: 0.7987 - val_loss: 0.5745 - val_acc: 0.6957
Epoch 16/40
5/5 [==============================] - 0s 19ms/step - loss: 0.3843 - acc: 0.8616 - val_loss: 0.4933 - val_acc: 0.7681
Epoch 17/40
```

```
5/5 [==============================] - 0s 20ms/step - loss: 0.3407 - acc: 0.8931 - v
al_loss: 0.4576 - val_acc: 0.7971
Epoch 18/40
5/5 [==============================] - 0s 20ms/step - loss: 0.2703 - acc: 0.9371 - v
al_loss: 0.4704 - val_acc: 0.7826
Epoch 19/40
5/5 [==============================] - 0s 19ms/step - loss: 0.2951 - acc: 0.9245 - v
al_loss: 0.5564 - val_acc: 0.6957
Epoch 20/40
5/5 [==============================] - 0s 21ms/step - loss: 0.2203 - acc: 0.9623 - v
al_loss: 0.4480 - val_acc: 0.7971
Epoch 21/40
5/5 [==============================] - 0s 21ms/step - loss: 0.2108 - acc: 0.9686 - v
al_loss: 0.4232 - val_acc: 0.8116
Epoch 22/40
5/5 [==============================] - 0s 21ms/step - loss: 0.1577 - acc: 0.9811 - v
al_loss: 0.4747 - val_acc: 0.7971
Epoch 23/40
5/5 [==============================] - 0s 20ms/step - loss: 0.1390 - acc: 0.9811 - v
al_loss: 0.4979 - val_acc: 0.7971
Epoch 24/40
5/5 [==============================] - 0s 20ms/step - loss: 0.1272 - acc: 0.9811 - v
al_loss: 0.5181 - val_acc: 0.7971
Epoch 25/40
5/5 [==============================] - 0s 19ms/step - loss: 0.1185 - acc: 0.9811 - v
al_loss: 0.5432 - val_acc: 0.7971
Epoch 26/40
5/5 [==============================] - 0s 21ms/step - loss: 0.1113 - acc: 0.9811 - v
al_loss: 0.5786 - val_acc: 0.7826
Epoch 27/40
5/5 [==============================] - 0s 21ms/step - loss: 0.1059 - acc: 0.9811 - v
al_loss: 0.6200 - val_acc: 0.7826
Epoch 28/40
5/5 [==============================] - 0s 21ms/step - loss: 0.1018 - acc: 0.9811 - v
al_loss: 0.6957 - val_acc: 0.7681
Epoch 29/40
5/5 [==============================] - 0s 21ms/step - loss: 0.1014 - acc: 0.9811 - v
al_loss: 0.6209 - val_acc: 0.7971
Epoch 30/40
5/5 [==============================] - 0s 21ms/step - loss: 0.0684 - acc: 0.9874 - v
al_loss: 0.7451 - val_acc: 0.7826
Epoch 31/40
5/5 [==============================] - 0s 20ms/step - loss: 0.0669 - acc: 0.9874 - v
al_loss: 0.7669 - val_acc: 0.7826
Epoch 32/40
5/5 [==============================] - 0s 21ms/step - loss: 0.0650 - acc: 0.9874 - v
al_loss: 0.7579 - val_acc: 0.7826
Epoch 33/40
5/5 [==============================] - 0s 24ms/step - loss: 0.0643 - acc: 0.9874 - v
al_loss: 0.7415 - val_acc: 0.7971
Epoch 34/40
5/5 [==============================] - 0s 21ms/step - loss: 0.0637 - acc: 0.9874 - v
al_loss: 0.7424 - val_acc: 0.7971
Epoch 35/40
5/5 [==============================] - 0s 19ms/step - loss: 0.0632 - acc: 0.9874 - v
al_loss: 0.7867 - val_acc: 0.7971
Epoch 36/40
5/5 [==============================] - 0s 23ms/step - loss: 0.0631 - acc: 0.9874 - v
al_loss: 0.7030 - val_acc: 0.8116
Epoch 37/40
5/5 [==============================] - 0s 20ms/step - loss: 0.0631 - acc: 0.9874 - v
al_loss: 0.6257 - val_acc: 0.8406
Epoch 38/40
5/5 [==============================] - 0s 23ms/step - loss: 0.0629 - acc: 0.9874 - v
```

```
al_loss: 0.6350 - val_acc: 0.8406
Epoch 39/40
5/5 [==============================] - 0s 24ms/step - loss: 0.0629 - acc: 0.9874 - v
al_loss: 0.6403 - val_acc: 0.8406
Epoch 40/40
5/5 [==============================] - 0s 23ms/step - loss: 0.0627 - acc: 0.9874 - v
al_loss: 0.6409 - val_acc: 0.8551
```

In [197…

```
df_lstm_score = score_df(history)
eval_plot(df_lstm_score)
```



Stacking Recurrent Layer

In [202…

```
model4 = Sequential()
model4.add(layers.Embedding(1600, 64, input_length= 50))
model4.add(layers.GRU(32, dropout=0.1, recurrent_dropout=0.5, return_sequences=True
model4.add(layers.GRU(64, dropout=0.1, recurrent_dropout=0.5))
model4.add(layers.Dense(2, activation='sigmoid'))

model4.compile(optimizer=RMSprop(), loss='binary_crossentropy', metrics=['acc'])
model4.summary()
history = model4.fit(train_input, Y_matrix, epochs=40, batch_size=32,
                     validation_split = 0.3)
```

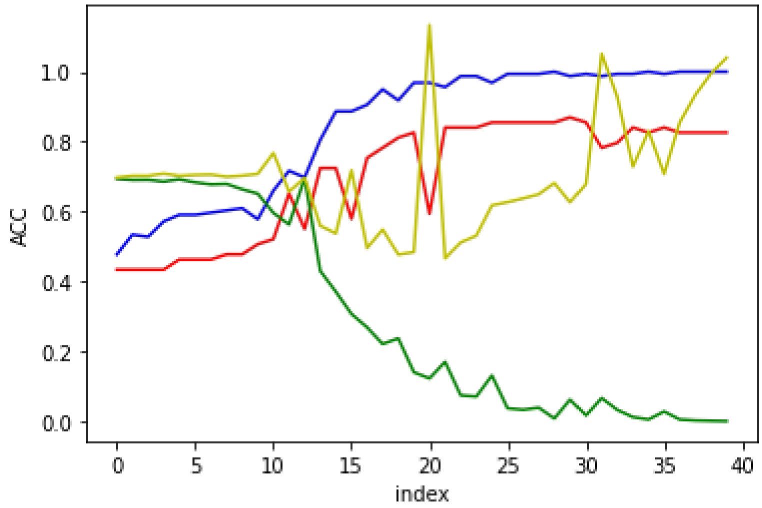
```
Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_15 (Embedding)    (None, 50, 64)            102400

 gru_6 (GRU)                 (None, 50, 32)            9408

 gru_7 (GRU)                 (None, 64)                18816

 dense_12 (Dense)            (None, 2)                 130

=================================================================
Total params: 130,754
Trainable params: 130,754
Non-trainable params: 0
_________________________________________________________________
Epoch 1/40
5/5 [==============================] - 7s 309ms/step - loss: 0.6943 - acc: 0.4780 -
val_loss: 0.6975 - val_acc: 0.4348
Epoch 2/40
5/5 [==============================] - 0s 62ms/step - loss: 0.6909 - acc: 0.5346 - v
al_loss: 0.7026 - val_acc: 0.4348
Epoch 3/40
5/5 [==============================] - 0s 65ms/step - loss: 0.6913 - acc: 0.5283 - v
al_loss: 0.7025 - val_acc: 0.4348
Epoch 4/40
5/5 [==============================] - 0s 63ms/step - loss: 0.6862 - acc: 0.5723 - v
al_loss: 0.7090 - val_acc: 0.4348
Epoch 5/40
5/5 [==============================] - 0s 65ms/step - loss: 0.6928 - acc: 0.5912 - v
al_loss: 0.7026 - val_acc: 0.4638
Epoch 6/40
5/5 [==============================] - 0s 63ms/step - loss: 0.6838 - acc: 0.5912 - v
al_loss: 0.7051 - val_acc: 0.4638
Epoch 7/40
5/5 [==============================] - 0s 60ms/step - loss: 0.6781 - acc: 0.5975 - v
al_loss: 0.7062 - val_acc: 0.4638
Epoch 8/40
5/5 [==============================] - 0s 64ms/step - loss: 0.6795 - acc: 0.6038 - v
al_loss: 0.7000 - val_acc: 0.4783
Epoch 9/40
5/5 [==============================] - 0s 64ms/step - loss: 0.6642 - acc: 0.6101 - v
al_loss: 0.7030 - val_acc: 0.4783
Epoch 10/40
5/5 [==============================] - 0s 64ms/step - loss: 0.6516 - acc: 0.5786 - v
al_loss: 0.7081 - val_acc: 0.5072
Epoch 11/40
5/5 [==============================] - 0s 61ms/step - loss: 0.5967 - acc: 0.6604 - v
al_loss: 0.7678 - val_acc: 0.5217
Epoch 12/40
5/5 [==============================] - 0s 64ms/step - loss: 0.5636 - acc: 0.7170 - v
al_loss: 0.6570 - val_acc: 0.6522
Epoch 13/40
5/5 [==============================] - 0s 63ms/step - loss: 0.6957 - acc: 0.6981 - v
al_loss: 0.6968 - val_acc: 0.5507
Epoch 14/40
5/5 [==============================] - 0s 61ms/step - loss: 0.4312 - acc: 0.8050 - v
al_loss: 0.5601 - val_acc: 0.7246
Epoch 15/40
5/5 [==============================] - 0s 65ms/step - loss: 0.3717 - acc: 0.8868 - v
al_loss: 0.5378 - val_acc: 0.7246
Epoch 16/40
5/5 [==============================] - 0s 60ms/step - loss: 0.3082 - acc: 0.8868 - v
```

```
al_loss: 0.7183 - val_acc: 0.5797
Epoch 17/40
5/5 [==============================] - 0s 57ms/step - loss: 0.2700 - acc: 0.9057 - v
al_loss: 0.4962 - val_acc: 0.7536
Epoch 18/40
5/5 [==============================] - 0s 57ms/step - loss: 0.2223 - acc: 0.9497 - v
al_loss: 0.5490 - val_acc: 0.7826
Epoch 19/40
5/5 [==============================] - 0s 57ms/step - loss: 0.2385 - acc: 0.9182 - v
al_loss: 0.4777 - val_acc: 0.8116
Epoch 20/40
5/5 [==============================] - 0s 57ms/step - loss: 0.1416 - acc: 0.9686 - v
al_loss: 0.4846 - val_acc: 0.8261
Epoch 21/40
5/5 [==============================] - 0s 59ms/step - loss: 0.1241 - acc: 0.9686 - v
al_loss: 1.1333 - val_acc: 0.5942
Epoch 22/40
5/5 [==============================] - 0s 57ms/step - loss: 0.1713 - acc: 0.9560 - v
al_loss: 0.4671 - val_acc: 0.8406
Epoch 23/40
5/5 [==============================] - 0s 59ms/step - loss: 0.0757 - acc: 0.9874 - v
al_loss: 0.5122 - val_acc: 0.8406
Epoch 24/40
5/5 [==============================] - 0s 58ms/step - loss: 0.0722 - acc: 0.9874 - v
al_loss: 0.5315 - val_acc: 0.8406
Epoch 25/40
5/5 [==============================] - 0s 57ms/step - loss: 0.1323 - acc: 0.9686 - v
al_loss: 0.6183 - val_acc: 0.8551
Epoch 26/40
5/5 [==============================] - 0s 58ms/step - loss: 0.0386 - acc: 0.9937 - v
al_loss: 0.6274 - val_acc: 0.8551
Epoch 27/40
5/5 [==============================] - 0s 62ms/step - loss: 0.0345 - acc: 0.9937 - v
al_loss: 0.6382 - val_acc: 0.8551
Epoch 28/40
5/5 [==============================] - 0s 60ms/step - loss: 0.0405 - acc: 0.9937 - v
al_loss: 0.6499 - val_acc: 0.8551
Epoch 29/40
5/5 [==============================] - 0s 60ms/step - loss: 0.0096 - acc: 1.0000 - v
al_loss: 0.6819 - val_acc: 0.8551
Epoch 30/40
5/5 [==============================] - 0s 57ms/step - loss: 0.0634 - acc: 0.9874 - v
al_loss: 0.6272 - val_acc: 0.8696
Epoch 31/40
5/5 [==============================] - 0s 63ms/step - loss: 0.0185 - acc: 0.9937 - v
al_loss: 0.6792 - val_acc: 0.8551
Epoch 32/40
5/5 [==============================] - 0s 56ms/step - loss: 0.0678 - acc: 0.9874 - v
al_loss: 1.0511 - val_acc: 0.7826
Epoch 33/40
5/5 [==============================] - 0s 55ms/step - loss: 0.0342 - acc: 0.9937 - v
al_loss: 0.9275 - val_acc: 0.7971
Epoch 34/40
5/5 [==============================] - 0s 58ms/step - loss: 0.0137 - acc: 0.9937 - v
al_loss: 0.7287 - val_acc: 0.8406
Epoch 35/40
5/5 [==============================] - 0s 57ms/step - loss: 0.0068 - acc: 1.0000 - v
al_loss: 0.8303 - val_acc: 0.8261
Epoch 36/40
5/5 [==============================] - 0s 58ms/step - loss: 0.0296 - acc: 0.9937 - v
al_loss: 0.7075 - val_acc: 0.8406
Epoch 37/40
5/5 [==============================] - 0s 60ms/step - loss: 0.0067 - acc: 1.0000 - v
al_loss: 0.8550 - val_acc: 0.8261
```

```
Epoch 38/40
5/5 [==============================] - 0s 59ms/step - loss: 0.0039 - acc: 1.0000 - v
al_loss: 0.9338 - val_acc: 0.8261
Epoch 39/40
5/5 [==============================] - 0s 60ms/step - loss: 0.0029 - acc: 1.0000 - v
al_loss: 0.9945 - val_acc: 0.8261
Epoch 40/40
5/5 [==============================] - 0s 57ms/step - loss: 0.0019 - acc: 1.0000 - v
al_loss: 1.0397 - val_acc: 0.8261
```

In [203…

```
df_lstm_score = score_df(history)
eval_plot(df_lstm_score)
```



BERT

- Bidirectional Encoder Representations from Transformers
- **Encoder**를 탑재하고있는 모델

- Transformer : **Encoder**(언어 -> 컴퓨터) + Decoder (컴퓨터 -> 언어) (생성모델)

구성

1. WordPiece : 문장 내 단어에 대한 Tokenizing + Padding / 빈도수가 낮은 단어를 서브 워드로 분리하여 분석 -> I am rewriting the posts -> [i , am, rewriting, the, posts] -> [i , am, re, ##write, ##ing, the, posts, ##s] -> 더 정확한 문맥 파악 / 본래 단어 복원 쉽게 진행

2. Contextual Embedding : 단어 앞/뒤에 태그를 부착하여, 순차적으로 벡터와 하여 문맥이 반영되게 끔 임베딩

3. Self Attention : 임베딩으로 벡터화된 단어들을 이용해 유사도를 계산하는 함수 (유사도가 반영된 가중함수를 연산 / 행렬 연산 + 활성 함수)

4. Pre-Training : 사전 학습

- 양방향으로 문맥의 모든 단어를 참조하여 예측하는 형태
- MLM (Masked Language Model) + Next Sentence Prediction (NSP)
- Masked Language Model : 입력받은 문장의 일부를 Random Masking -> 모델이 문맥을 통해 본래의 단어를 예측하도록 구성 (문장 내 단어 처리)
- Next Sentence Prediction (NSP) : 두개의 문장이 서로 이어지는 문장인지를 판별 (문장 간 처리)

5. Fine-Tuninig : 사전학습된 모델에 일부 데이터만 추가로 학습하여 모델을 구성 방식

In [208…
```
import tensorflow as tf
import torch
```

In [210…
```
df1 = pd.read_csv('company_Train_data.csv')
df1['Target'] = df1['기업성장여부'].replace({'성장':1, '정체':0})
```

In [212…
```
X = df1[['한줄평']]
Y = df1['Target'].values
```

In [216…
```
# Contextual Embedding 사전 작업 / [CLS] + [SEP]
X_contextual = '[CLS] '+ X + ' [SEP]'
```

In [218…
```
# !pip insatll --user transformers
```

In [217…
```
# WordPiece
from transformers import BertTokenizer
```

In [220…
```
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased',
                                          do_lower_case=False)
```

In [224…
```
X_tarin_list = [ ]
for i in range(0, len(X_contextual)):
    word_piece = tokenizer.tokenize( X_contextual.values.tolist()[i][0] )
    X_tarin_list.append(word_piece)
```

In [233…
```
# Padding
X_sequence = [ ]
for k in range(0, len(X_tarin_list)):
    token_seq = tokenizer.convert_tokens_to_ids(X_tarin_list[k])
    X_sequence.append(token_seq) # id 값만 부여

print(pd.DataFrame(X_sequence).shape) # 가장 긴 문장에 맞춰서 Padding

X_padding =  pad_sequences(X_sequence, maxlen=300, padding='post')
```

```
(304, 300)
```

In [238…
```
# Attention Mask [1,23,40, 0,0,0,0,0,0]
# 패딩작업의 결과로 생성된 불필요한 0 값을 구분지어주는 리스트를 형성
X_atmask = [ ]
for j in X_padding:
    X_atmask.append( [float(k>0) for k in j ] )
    # 단어가 있는 부분은 1 / 없는 부분은 0
    # 불필요한 0 값을 학습에 제외하고 학습을 수행하기 위해 at mask 제작
```

In [239…
```
# 학습데이터 검증데이터 분할
X_train, X_test, Y_train, Y_test = train_test_split(X_padding, Y, random_state=1234)
# Attenion Mask 분할
X_train_mask, X_test_mask, _, _ = train_test_split(X_atmask,X_padding,random_state=1
```

In [242…
```
# 3가지의 행렬을 하나의 텐서행렬로 병합 ( 패딩된 X / 마스크 X / Y )
X_train_tensor = torch.tensor(X_train)
Y_train_tensor = torch.tensor(Y_train)
X_train_mask_tensor = torch.tensor(X_train_mask)

X_test_tensor = torch.tensor(X_test)
Y_test_tensor = torch.tensor(Y_test)
X_test_mask_tensor = torch.tensor(X_test_mask)
```

In [241…
```
from torch.utils.data import TensorDataset
from torch.utils.data import DataLoader, RandomSampler, SequentialSampler
```

In [243…
```
train_data = TensorDataset(X_train_tensor, X_train_mask_tensor, Y_train_tensor)
test_data  = TensorDataset(X_test_tensor, X_test_mask_tensor, Y_test_tensor)
```

In [244…
```
train_sampler = RandomSampler(train_data)
test_sampler  = SequentialSampler(test_data)

train_dataloader = DataLoader(train_data, sampler= train_sampler, batch_size= 32)
test_dataloader  = DataLoader(test_data,  sampler= test_sampler, batch_size= 32)
```

BERT Model 호출 및 학습

In [247…
```
# Pre Training Model을 호출
from transformers import BertForSequenceClassification, AdamW, BertConfig
```

In [248…
```
model = BertForSequenceClassification.from_pretrained("bert-base-multilingual-cased"
                                                      num_labels=2)
model.cpu( )
```

```
Some weights of the model checkpoint at bert-base-multilingual-cased were not used w
hen initializing BertForSequenceClassification: ['cls.predictions.transform.LayerNor
m.weight', 'cls.seq_relationship.bias', 'cls.predictions.transform.dense.weight', 'c
ls.predictions.decoder.weight', 'cls.seq_relationship.weight', 'cls.predictions.tran
sform.LayerNorm.bias', 'cls.predictions.transform.dense.bias', 'cls.predictions.bia
s']
- This IS expected if you are initializing BertForSequenceClassification from the ch
eckpoint of a model trained on another task or with another architecture (e.g. initi
alizing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassification from th
e checkpoint of a model that you expect to be exactly identical (initializing a Bert
ForSequenceClassification model from a BertForSequenceClassification model).
Some weights of BertForSequenceClassification were not initialized from the model ch
eckpoint at bert-base-multilingual-cased and are newly initialized: ['classifier.bia
s', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for
predictions and inference.
```

Out[248]:
```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(119547, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (1): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (2): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
```

```
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (3): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (4): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
```

```
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (5): BertLayer(
      (attention): BertAttention(
        (self): BertSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (6): BertLayer(
      (attention): BertAttention(
        (self): BertSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (7): BertLayer(
      (attention): BertAttention(
        (self): BertSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
```

```
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (8): BertLayer(
      (attention): BertAttention(
        (self): BertSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (9): BertLayer(
      (attention): BertAttention(
        (self): BertSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (10): BertLayer(
      (attention): BertAttention(
        (self): BertSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
```

```
                (value): Linear(in_features=768, out_features=768, bias=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
              (output): BertSelfOutput(
                (dense): Linear(in_features=768, out_features=768, bias=True)
                (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
            )
            (intermediate): BertIntermediate(
              (dense): Linear(in_features=768, out_features=3072, bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): BertOutput(
              (dense): Linear(in_features=3072, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (11): BertLayer(
            (attention): BertAttention(
              (self): BertSelfAttention(
                (query): Linear(in_features=768, out_features=768, bias=True)
                (key): Linear(in_features=768, out_features=768, bias=True)
                (value): Linear(in_features=768, out_features=768, bias=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
              (output): BertSelfOutput(
                (dense): Linear(in_features=768, out_features=768, bias=True)
                (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
            )
            (intermediate): BertIntermediate(
              (dense): Linear(in_features=768, out_features=3072, bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): BertOutput(
              (dense): Linear(in_features=3072, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
        )
      )
      (pooler): BertPooler(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (activation): Tanh()
      )
    )
    (dropout): Dropout(p=0.1, inplace=False)
    (classifier): Linear(in_features=768, out_features=2, bias=True)
  )
```

In [252…

```
# fine-Tuning
# 최적화 파라미터 설정
optimizer = AdamW(model.parameters(), lr= 2e-5, eps=1e-8) # lr= 2e-5, eps=1e-8
epochs = 4
total_steps = len(train_dataloader) * 4
```

In [253…

```
# Scheduler  Setting
from transformers import get_linear_schedule_with_warmup
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0,
                                            num_training_steps=total_steps)
```

학습 수행

In [254…
```
def flat_accuracy(preds, labels):
    pred_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return np.sum(pred_flat == labels_flat) / len(labels_flat)

def format_time(elapsed):
    elapsed_rounded = int(round((elapsed)))
    return str(datetime.timedelta(seconds=elapsed_rounded))
```

In [256…
```
import random
import time
import datetime
```

In [ ]:
```
#랜덤시드 고정
seed_val = 42
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)

#그래디언트 초기화
model.zero_grad()

# 학습
for epoch_i in range(0, epochs):
    print("")
    print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
    print('Training...')

    # 시작 시간 설정
    t0 = time.time()

    # 로스 초기화
    total_loss = 0

    # 훈련모드로 변경
    model.train()

    # 데이터로더에서 배치만큼 반복하여 가져옴
    for step, batch in enumerate(train_dataloader):
        # 경과 정보 표시
        if step % 500 == 0 and not step == 0:
            elapsed = format_time(time.time() - t0)
            print('  Batch {:>5,}  of  {:>5,}.    Elapsed: {:}.'.format(step, len(tr

        # 배치에서 데이터 추출
        b_input_ids, b_input_mask, b_labels = batch

        # Forward 수행
        outputs = model(b_input_ids,
                        token_type_ids=None,
                        attention_mask=b_input_mask,
                        labels=b_labels)

        # 로스 구함
        loss = outputs[0]

        # 총 로스 계산
        total_loss += loss.item()
```

```
        # Backward 수행으로 그래디언트 계산
        loss.backward()

        # 그래디언트 클리핑
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

        # 그래디언트를 통해 가중치 파라미터 업데이트
        optimizer.step()

        # 스케줄러로 학습률 감소
        scheduler.step()

        # 그래디언트 초기화
        model.zero_grad()

    # 평균 로스 계산
    avg_train_loss = total_loss / len(train_dataloader)

    print("")
    print("  Average training loss: {0:.2f}".format(avg_train_loss))
    print("  Training epcoh took: {:}".format(format_time(time.time() - t0)))
####################################################################################
    print("")
    print("Running Validation...")

    #시작 시간 설정
    t0 = time.time()

    # 평가모드로 변경
    model.eval()

    # 변수 초기화
    eval_loss, eval_accuracy = 0, 0
    nb_eval_steps, nb_eval_examples = 0, 0

    # 데이터로더에서 배치만큼 반복하여 가져옴
    for batch in validation_dataloader:
        # 배치를 GPU에 넣음
        batch = tuple(t.to(device) for t in batch)

        # 배치에서 데이터 추출
        b_input_ids, b_input_mask, b_labels = batch

        # 그래디언트 계산 안함
        with torch.no_grad():
            # Forward 수행
            outputs = model(b_input_ids,
                            token_type_ids=None,
                            attention_mask=b_input_mask)

        # 로스 구함
        logits = outputs[0]

        # CPU로 데이터 이동
        logits = logits.detach().cpu().numpy()
        label_ids = b_labels.to('cpu').numpy()

        # 출력 로짓과 라벨을 비교하여 정확도 계산
        tmp_eval_accuracy = flat_accuracy(logits, label_ids)
        eval_accuracy += tmp_eval_accuracy
        nb_eval_steps += 1
```

```
    print("  Accuracy: {0:.2f}".format(eval_accuracy/nb_eval_steps))
    print("  Validation took: {:}".format(format_time(time.time() - t0)))
```

```
======== Epoch 1 / 4 ========
Training...
```