In [ ]:
```
!pip install transformers
```

In [2]:
```
import pandas as pd
import os
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive/')
```

In [ ]:
```
!git clone https://github.com/hyejeong1019/Sentiment_Data.git
```

In [ ]:
```
os.listdir('Sentiment_Data')
```

In [78]:
```
data_path = "Sentiment_Data/"
train_data = pd.read_csv(data_path + "train_data_prep_sent.csv")
test_data = pd.read_csv(data_path + "test_data_prep_sent.csv")
```

In [ ]:
```
train_data[:5]
```

In [ ]:
```
test_data.head()
```

In [44]:
```
import numpy as np
DATA_COL = "document"  # 입력 문장을 포함하고 있는 칼럼
LABEL_COL = "label"    # 긍정인지 부정인지를 (1=긍정,0=부정) 포함하고 있는 칼럼
X_train_text = list(train_data[DATA_COL][:10000])
y_train = np.array(train_data[LABEL_COL][:10000])
X_test_text = list(test_data[DATA_COL][:2000])
y_test = np.array(test_data[LABEL_COL][:2000])
```

In [ ]:
```
len(X_train_text), len(y_train),len(X_test_text),  len(y_test)
```

In [34]:
```
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
```

In [ ]:
```
BERT_SEQ_LEN = 128 #SEQ_LEN : 버트에 들어갈 인풋의 길이
from tqdm import tqdm
import numpy as np
def convert_Bert_input_data(sentence_list):
    global tokenizer
    tokens, masks, segments, targets = [], [], [], []

    for sentence in tqdm(sentence_list):
        token = tokenizer.encode(sentence, max_length=BERT_SEQ_LEN, truncation=True
        num_zeros = token.count(0) # 마스크는 토큰화한 문장에서 패딩이 아닌 부분은 1
        mask = [1] * (BERT_SEQ_LEN - num_zeros) + [0] * num_zeros
        segment = [0] * BERT_SEQ_LEN # 문장의 전후관계를 구분해주는 세그먼트는 문장0

        # 버트 인풋으로 들어가는 token, mask, segment를 tokens, segments에 각각 저장
        tokens.append(token)
        masks.append(mask)
        segments.append(segment)

    # tokens, masks, segments를 numpy array로 지정
    return [np.array(tokens), np.array(masks), np.array(segments)]

X_train = convert_Bert_input_data(X_train_text)
```

In [ ]:
```
print(list(map(len,X_train)))
```

In [ ]:
```
X_test = convert_Bert_input_data(X_test_text)
```

In [ ]:
```
print(list(map(len,X_test)))
```

In [ ]:
```
import tensorflow as tf

# TPU 객체 지정
TPU = True
if TPU:
  resolver = tf.distribute.cluster_resolver.TPUClusterResolver(tpu='grpc://' + os.en
  tf.config.experimental_connect_to_cluster(resolver)
  tf.tpu.experimental.initialize_tpu_system(resolver)
else:
  pass
```

In [ ]:
```
# Rectified Adam 옵티마이저 사용
!pip install tensorflow_addons
```

In [ ]:
```
import tensorflow_addons as tfa
opt = tfa.optimizers.RectifiedAdam(lr=1.0e-5, weight_decay=0.0025, warmup_proportion
```

In [42]:
```
from transformers import TFBertModel

def create_sentiment_bert():
  # 버트 pretrained 모델 로드
  model = TFBertModel.from_pretrained('bert-base-multilingual-cased')
  # 토큰 인풋, 마스크 인풋, 세그먼트 인풋 정의
  token_inputs = tf.keras.layers.Input((BERT_SEQ_LEN,), dtype=tf.int32, name='input_
  mask_inputs = tf.keras.layers.Input((BERT_SEQ_LEN,), dtype=tf.int32, name='input_m
  segment_inputs = tf.keras.layers.Input((BERT_SEQ_LEN,), dtype=tf.int32, name='inpu
  # 인풋이 [토큰, 마스크, 세그먼트]인 모델 정의
  bert_outputs = model([token_inputs, mask_inputs, segment_inputs])

  bert_outputs = bert_outputs[1]
  sentiment_first = tf.keras.layers.Dense(1, activation='sigmoid', kernel_initialize
  sentiment_model = tf.keras.Model([token_inputs, mask_inputs, segment_inputs], sent

  sentiment_model.compile(optimizer=opt, loss=tf.keras.losses.BinaryCrossentropy(),
  return sentiment_model
```

In [ ]:
```
# TPU 실행 시
if TPU:
  strategy = tf.distribute.TPUStrategy(resolver)
  # 함수를 strategy.scope로 묶어 줌
  with strategy.scope():
    sentiment_model = create_sentiment_bert()
  #sentiment_model.fit(X_train, y_train, epochs=4, shuffle=True, batch_size=100, val
else:
  # GPU 모드로 훈련시킬 때
  sentiment_model = create_sentiment_bert()
sentiment_model.fit(X_train, y_train, epochs=4, shuffle=True, batch_size=100, valid
```

In [46]:
```
# 학습된 모델을 저장할 foler 지정 (해당 폴더가 없으면 만들어 놓기)
model_path = "gdrive/My Drive/Colab Notebooks/Sentiment_Model/"
```

In [47]:
```
sentiment_model.save_weights(model_path+"huggingface_bert_32.h5")
```

In [ ]:
```
#TPU를 사용하기 위해서
with strategy.scope():
```

```
  result = sentiment_model.evaluate(X_test, y_test)
```

In [ ]:
```
result
```

In [71]:
```
def analyze_sentiment(sentence):
    sentence_fv = convert_Bert_input_data(sentence)
    result = sentiment_model.predict(sentence_fv)
    result = np.ravel(result)
    if result[1] > 0.5:
      print(f"(긍정 확률 : {result[1]*100:.2f}%) 긍정적인 리뷰입니다.")
    else:
      print(f"(부정 확률 : {(1-result[1])*100:.2f}%) 부정적인 리뷰입니다.")
```

In [ ]:
```
analyze_sentiment("이 영화 개꿀잼 ㅋㅋㅋ ")
```

In [ ]:
```
analyze_sentiment("이 영화 핵노잼 ㅠㅠ")
```

In [ ]:
```
analyze_sentiment("이딴게 영화냐 ㅉㅉ")
```

In [ ]:
```
analyze_sentiment("감독 뭐하는 놈이냐?")
```

In [ ]:
```
analyze_sentiment("와 개쩐다 정말 세계관 최강자들의 영화다")
```