```
%tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

텍스트 분류

copied and modified from https://www.tensorflow.org/tutorials/keras/text_classification

더블클릭 또는 Enter 키를 눌러 수정

```
## 설정
VOCA_SIZE = 4000 # 어휘 사전의 크기
EMBEDDING_SIZE = 64 # 단어를 임베딩한 벡터 크기 EunAh:한 word를 몇개의 숫자로 나타낼 것인지
```

데이터 로딩

EunAh: 문장을 읽어서 긍정인지 부정인지 이해하는 과정

```
import tensorflow as tf

print('Loading data...')
(train_x, train_y), (test_x, test_y) = tf.keras.datasets.imdb.load_data(num_words=VOCA_SIZE)

print(train_x.shape)
print(train_y.shape)
print(test_x.shape)
print(test_y.shape)
```

```
Loading data...
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17465344/17464789 [==============================] - 0s 0us/step
(25000,)
(25000,)
(25000,)
(25000,)
```

```
print(type(train_x[0])) #data type을 확인
```

```
<class 'list'>
```

데이터 보기

```
print(train_x[:5])
print(train_y[:5])
```

```
[list([1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 2, 66, 3941, 4, 173, 36, 256, 5, 25,
 list([1, 194, 1153, 194, 2, 78, 228, 5, 6, 1463, 2, 2, 134, 26, 4, 715, 8, 118, 1634, 14, 39
 list([1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 2, 54, 61, 369, 13, 71, 149, 14, 22, 112,
 list([1, 4, 2, 2, 33, 2804, 4, 2040, 432, 111, 153, 103, 4, 1494, 13, 70, 131, 67, 11, 61, 2
 list([1, 249, 1323, 7, 61, 113, 10, 10, 13, 1637, 14, 20, 56, 33, 2401, 18, 457, 88, 13, 262
[1 0 0 1 0]
```

각 숫자가 각 워드에 해당 워드의 dictionary index. 되게 수렴이 느리고..

텍스트로 데이터 보기

```
# 단어와 정수 인덱스를 매핑한 딕셔너리
word_index = tf.keras.datasets.imdb.get_word_index()

# 처음 몇 개 인덱스는 사전에 정의되어 있습니다
word_index = {k:(v+3) for k,v in word_index.items()}
word_index["<PAD>"] = 0 #빈 자리를 채워넣는걸 0으로
word_index["<START>"] = 1
word_index["<UNK>"] = 2  # unknown
word_index["<UNUSED>"] = 3

reverse_word_index = dict([(value, key) for (key, value) in word_index.items()]) #숫자를 주면 그 숫

def decode_review(text):
    return ' '.join([reverse_word_index.get(i, '?') for i in text])

print(reverse_word_index)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_i
1646592/1641221 [==============================] - 0s 0us/step
{34704: 'fawn', 52009: 'tsukino', 52010: 'nunnery', 16819: 'sonja', 63954: 'vani', 1411: 'woo
```

```
print(decode_review(train_x[0]))
```

```
<START> this film was just brilliant casting location scenery story direction <UNK> really su
```

```
print(word_index)
```

```
{'fawn': 34704, 'tsukino': 52009, 'nunnery': 52010, 'sonja': 16819, 'vani': 63954, 'woods': 1
```

각 데이터의 길이

```
print(len(train_x[0]))
print(len(train_x[1]))
print(len(train_x[2]))
print(len(train_x[3]))
print(len(train_x[4]))
```

```
218
189
141
550
147
```

데이터 길이 일정하게 하기

```
print(train_x[0])
print(len(train_x[0]))
```

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 2, 66, 3941, 4, 173, 36, 256, 5, 25, 100,
218
```

```
from tensorflow.keras.preprocessing import sequence

train_x = sequence.pad_sequences(train_x, maxlen=400, padding='post')
test_x = sequence.pad_sequences(test_x, maxlen=400, padding='post')
print(train_x.shape)
print(test_x.shape)
```

```
(25000, 400)
(25000, 400)
```

```
print(train_x[0])
print(len(train_x[0]))
```

```
[    1   14   22   16   43  530  973 1622 1385   65  458    2   66 3941
     4  173   36  256    5   25  100   43  838  112   50  670    2    9
    35  480  284    5  150    4  172  112  167    2  336  385   39    4
   172    2 1111   17  546   38   13  447    4  192   50   16    6  147
  2025   19   14   22    4 1920    2  469    4   22   71   87   12   16
    43  530   38   76   15   13 1247    4   22   17  515   17   12   16
   626   18    2    5   62  386   12    8  316    8  106    5    4 2223
     2   16  480   66 3785   33    4  130   12   16   38  619    5   25
   124   51   36  135   48   25 1415   33    6   22   12  215   28   77
    52    5   14  407   16   82    2    8    4  107  117    2   15  256
     4    2    7 3766    5  723   36   71   43  530  476   26  400  317
    46    7    4    2 1029   13  104   88    4  381   15  297   98   32
  2071   56   26  141    6  194    2   18    4  226   22   21  134  476
    26  480    5  144   30    2   18   51   36   28  224   92   25  104
```

CNN 모델 사용

```
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

EunAh: Embedding이 들어간 것만 다름

```
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, Dropout, Activation
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D

model = Sequential()
model.add(Input(400)) #입력노드 갯수 400개
model.add(Embedding(VOCA_SIZE, EMBEDDING_SIZE)) # 텍스트는 임베딩 해서 사용한다.EunAh:400개가 64개의
model.add(Dropout(0.2))
model.add(Conv1D(250, 3))
model.add(GlobalMaxPooling1D())
model.add(Dense(250))
model.add(Dropout(0.2))
model.add(Activation('relu'))
model.add(Dense(1)) #출력노드 갯수 1개
model.add(Activation('sigmoid'))

model.summary()

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy']) #EunAh: 결과가 바

model.fit(train_x, train_y, batch_size=32, epochs=10, validation_data=(test_x, test_y))
```

RNN 모델 사용

CNN 사용할때는 한 epochs이 7초 걸렸음 RNN은 CNN보다 무지하게 느림- 10배 이상 느림 (recurrent하게 되돌아가기 때문)

```
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, Dropout, Activation
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D
from tensorflow.keras.layers import Bidirectional, LSTM

model = Sequential()
model.add(Input(400))
model.add(Embedding(VOCA_SIZE, EMBEDDING_SIZE))
model.add(Dropout(0.2))
# model.add(Conv1D(250, 3)) #CNN에 있는 code
# model.add(GlobalMaxPooling1D()) #CNN에 있는 code
model.add(Bidirectional(LSTM(64))) # ADD 사용하는 구체적인 내용은 Bidirectional 또는 RNN, LSTM 또는
model.add(Dense(250))
model.add(Dropout(0.2))
model.add(Activation('relu'))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.summary()

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(train_x, train_y, batch_size=32, epochs=10, validation_data=(test_x, test_y))
```

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/keras/initializer
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_vari
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 400, 64)           256000
_________________________________________________________________
dropout (Dropout)            (None, 400, 64)           0
_________________________________________________________________
conv1d (Conv1D)              (None, 398, 250)          48250
_________________________________________________________________
global_max_pooling1d (Global (None, 250)               0
_________________________________________________________________
dense (Dense)                (None, 250)               62750
_________________________________________________________________
dropout_1 (Dropout)          (None, 250)               0
_________________________________________________________________
activation (Activation)      (None, 250)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 251
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 367,251
Trainable params: 367,251
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/nn_impl.py:18
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 25000 samples, validate on 25000 samples
Epoch 1/10
25000/25000 [==============================] - 16s 648us/sample - loss: 0.4043 - acc: 0.7985
Epoch 2/10
25000/25000 [==============================] - 10s 402us/sample - loss: 0.2294 - acc: 0.9094
Epoch 3/10
25000/25000 [==============================] - 10s 401us/sample - loss: 0.1628 - acc: 0.9395
Epoch 4/10
25000/25000 [==============================] - 10s 401us/sample - loss: 0.1131 - acc: 0.9588
Epoch 5/10
25000/25000 [==============================] - 10s 402us/sample - loss: 0.0771 - acc: 0.9726
Epoch 6/10
```

```
loss, acc = model.evaluate(test_x, test_y)
print("loss =", loss)
print("acc =", acc)
```

```
25000/25000 [==============================] - 3s 131us/sample - loss: 0.5335 - acc: 0.8798
loss = 0.5335088743007184
acc = 0.87984
25000/25000 [==============================] - 10s 403us/sample - loss: 0.0294 - acc: 0.9896
```

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/init_ops.py:9
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/init_ops.py:9
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/init_ops.py:9
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 400, 64)           256000
_________________________________________________________________
dropout_2 (Dropout)          (None, 400, 64)           0
_________________________________________________________________
```

```
loss, acc = model.evaluate(test_x, test_y)
print("loss =", loss)
print("acc =", acc)
```

```
dropout_3 (Dropout)          (None, 250)               0
_________________________________________________________________
activation_2 (Activation)    (None, 250)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 251
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 354,549
Trainable params: 354,549
Non-trainable params: 0
_________________________________________________________________
Train on 25000 samples, validate on 25000 samples
Epoch 1/10
25000/25000 [==============================] - 1609s 64ms/sample - loss: 0.5058 - acc: 0.7510
Epoch 2/10
24992/25000 [============================>.] - ETA: 0s - loss: 0.3811 - acc: 0.8458
```