Natural Language Processing

(using IMDB dataset)

V2016120 김태형

Keras를 이용하여 IMDB dataset으로 NLP학습을 해보겠습니다.
IMDB dataset은 스탠포드 연구원에 의해 수집되었으며,
Large Movie Review Sentence dataset고, 문장에 따라 good(긍정) / bad(부정)으로 나뉘는 Label이 되어 있습니다.

Keras로 IMDB 데이터셋을 불러오면
training용 Data 25000 / test용 Data 25000로 구성이 되어있습니다.

```
from keras.datasets import imdb

#load the dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data()
```

word Embedding 이란?

- 자연어 처리 분야(NLP)에서 학습을 할 수 있도록, 텍스트를 숫자로 변환한 것으로 동일한 텍스트에 대해 다른 숫자로 표현이 가능하게
시켜주는 부분이 word Embedding이 주된 목적이며, 단어를 고차원 벡터로 매핑시켜주는 것 이며 수지를 입력 값으로 요구할 때
신경망과 관련된 자연어 처리 문제로 작업 할 때 아주 유용한 방법입니다.
Keras에서는 Embedding layer를 제공하며, 간편히 사용할 수 있습니다.

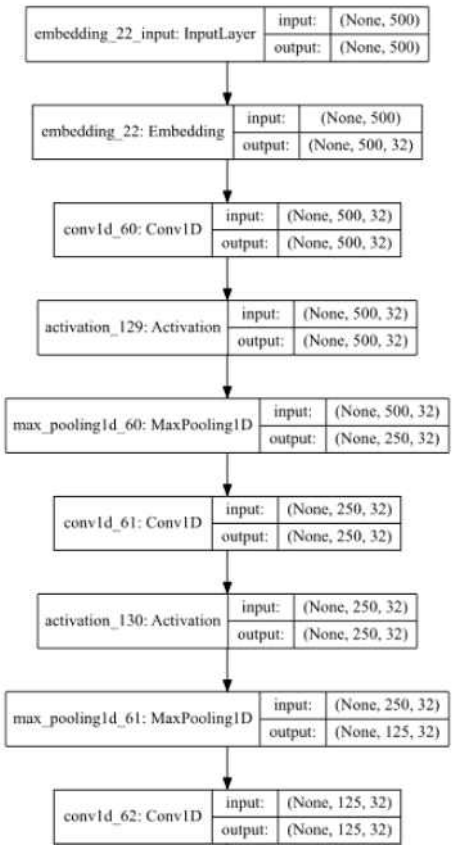
코드와 같이 만약 우리가 dataset에서 가장 많이 사용된 5,000개의 단어에만 관심이 있다고 가정을 한다면, 크기는 5,000이고
각 단어를 나타내기 위해서 32차원 벡터를 사용하도록 선택 할수 있으며, 문장의 최대 길이를 500단어로 제한하고 그보다 긴 리뷰는 잘라내고 0 값으로 채운 패딩 리뷰를 선택 할 수 있으며, 마지막으로 Embedding layer를 이용하면 32x 500인 matrix가 된다.

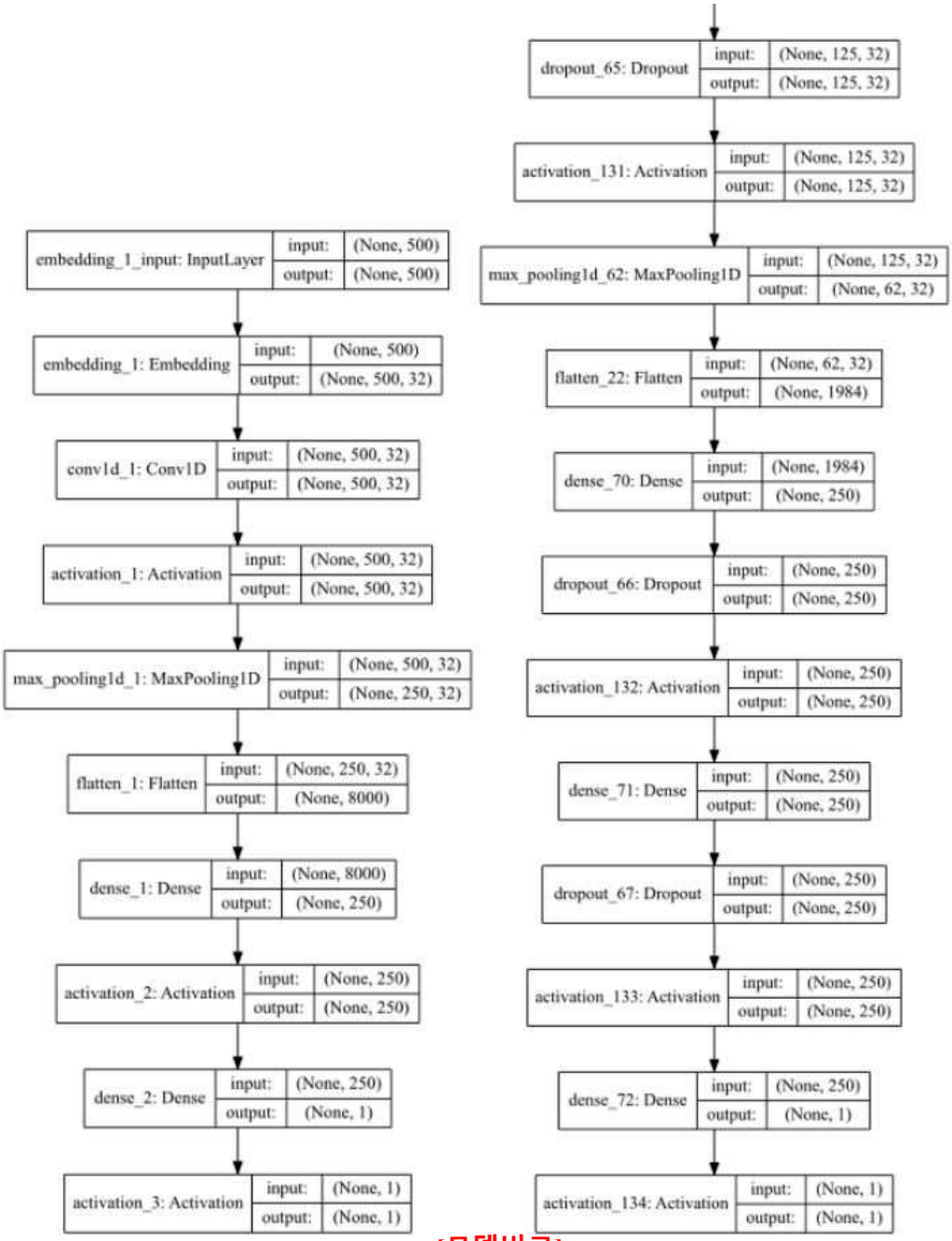
```
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence

imdb.load_data(nb_words=5000, test_split=0.33)
x_train = sequence.pad_sequences(x_train, maxlen=500)
x_test = sequence.pad_sequences(x_test, maxlen=500)

Embedding(5000, 32, input_length=500)
```

One-Dimensional Convolutional Neural Network 이란?

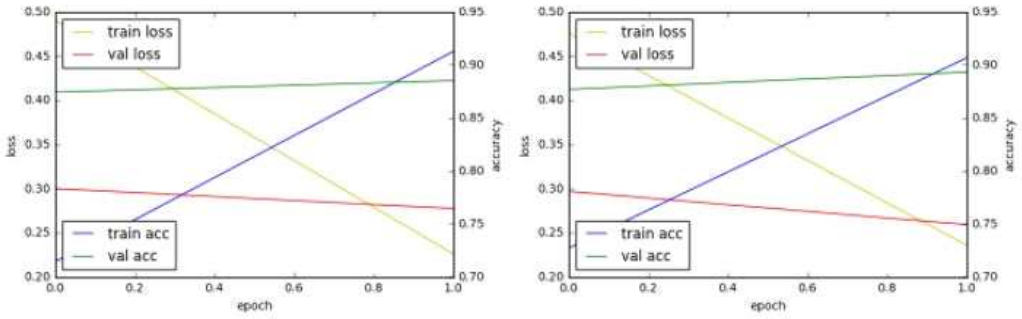
- CNN은 보통 이미지 인식 부분에서 위치와 방향에 견고하면서 이미지 데이터 공간 구조를 이해 하도록 보통 설계가 되었는데, IMDB(영화리뷰)에서도 단어의 1차원 시퀀스와 같은 스퀀스에 사용을 할 수 있고, MaxPooling도 사용을 할 수 있다.

지금 까지 했던 내용을 바탕으로 Keras로 모델을 구성하고 검증 정확률이 제일 높은 모델을 비교하며 테스트 해보겠습니다



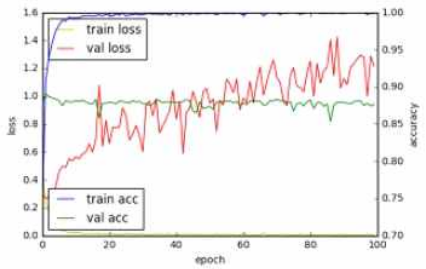

embedding_1_input: InputLayer | input: (None, 500) | output: (None, 500)
embedding_1: Embedding | input: (None, 500) | output: (None, 500, 32)
conv1d_1: Conv1D | input: (None, 500, 32) | output: (None, 500, 32)
activation_1: Activation | input: (None, 500, 32) | output: (None, 500, 32)
max_pooling1d_1: MaxPooling1D | input: (None, 500, 32) | output: (None, 250, 32)
flatten_1: Flatten | input: (None, 250, 32) | output: (None, 8000)
dense_1: Dense | input: (None, 8000) | output: (None, 250)
activation_2: Activation | input: (None, 250) | output: (None, 250)
dense_2: Dense | input: (None, 250) | output: (None, 1)
activation_3: Activation | input: (None, 1) | output: (None, 1)
dropout_65: Dropout | input: (None, 125, 32) | output: (None, 125, 32)
activation_131: Activation | input: (None, 125, 32) | output: (None, 125, 32)
max_pooling1d_62: MaxPooling1D | input: (None, 125, 32) | output: (None, 62, 32)
flatten_22: Flatten | input: (None, 62, 32) | output: (None, 1984)
dense_70: Dense | input: (None, 1984) | output: (None, 250)
dropout_66: Dropout | input: (None, 250) | output: (None, 250)
activation_132: Activation | input: (None, 250) | output: (None, 250)
dense_71: Dense | input: (None, 250) | output: (None, 250)
dropout_67: Dropout | input: (None, 250) | output: (None, 250)
activation_133: Activation | input: (None, 250) | output: (None, 250)
dense_72: Dense | input: (None, 250) | output: (None, 1)
activation_134: Activation | input: (None, 1) | output: (None, 1)

[모델비교]



[성능비교]

[88.51%] [89.34%]



[동일한 모델로 epoch수를 늘렸을 경우] ->Train Loss는 줄어들어 0에 가까워지는데, Validation Loss는 증가하는 모습을 보이며,
또한 정확도 역시 증아지지 않고 감소되는 overfitting 된 모습을 볼 수 있었다.