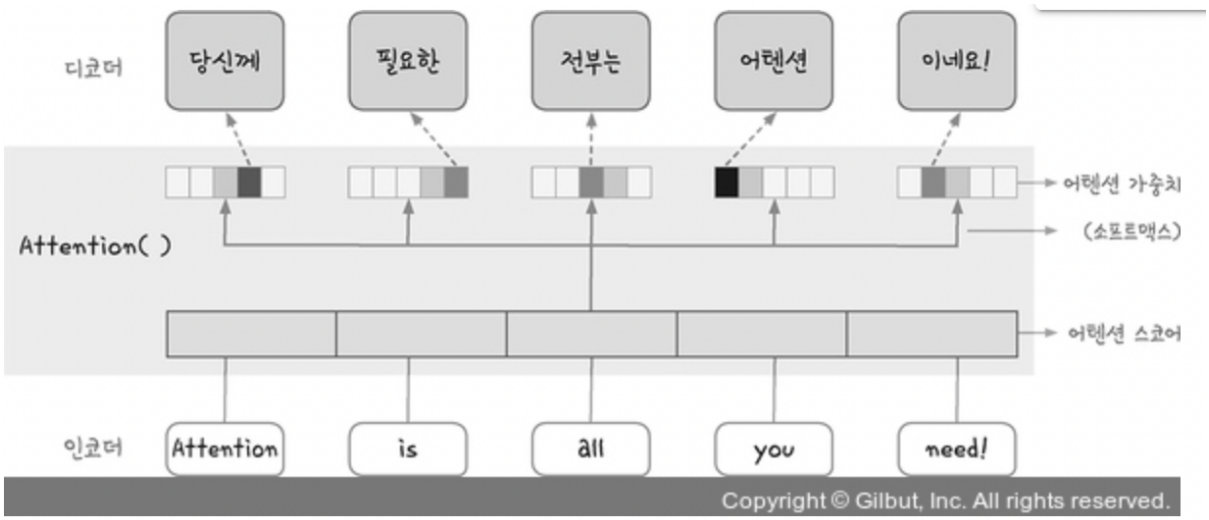
14주차 - 대면 Attention




```
!pip install attention

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Embedding, LSTM, Conv1
D, MaxPooling1D
from tensorflow.keras.datasets import imdb, reuters
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model, to_categorical
from attention import Attention

import numpy as np
import matplotlib.pyplot as plt

(X_train, y_train), (X_test, y_test) = reuters.load_data(num_words=1000, test_split=0.
2)

category = np.max(y_train) + 1
print(category, '카테고리')
print(len(X_train), '학습용 뉴스 기사')
print(len(X_test), '테스트용 뉴스 기사')

# 단어의 수를 맞춤
X_train = sequence.pad_sequences(X_train, maxlen=100)
X_test = sequence.pad_sequences(X_test, maxlen=100)

#데이터 원핫인코딩 형태로 변환 -> categorical data
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

```
model = Sequential()
model.add(Embedding(1000, 100))
model.add(Dropout(0.5))
model.add(LSTM(64, return_sequences=True, activation='tanh'))
model.add(Attention())
model.add(Dropout(0.5))
model.add(Dense(46, activation='softmax'))
model.summary()
```

Model: "sequential_22"

Layer (type) Output Shape Param

embedding_22 (Embedding) (None, None, 100) 100000

dropout_32 (Dropout) (None, None, 100) 0

lstm_21 (LSTM) (None, None, 64) 42240

attention_21 (Attention) (None, 128) 20480

dropout_33 (Dropout) (None, 128) 0

dense_20 (Dense) (None, 46) 5934

==========

Total params: 168,654
Trainable params: 168,654
Non-trainable params: 0

→ 잘라서 가중치를 주고 학습 시킴!!!!! -> encoding => Decoding == autoencoder와 유사함 (**LSTM -뒤에 Attention 모델 위치 => 자를 수 있도록 -> 3% 증가함 정확도)** bert -> 자연어처리모델

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=5)

history = model.fit(X_train, y_train, batch_size=20, epochs=200, validation_data=(X_tes
t, y_test), callbacks=[early_stopping_callback])

model.evaluate(X_test, y_test)[1]
```

0.73정도의 정확도를 가짐

```
# 학습셋과 테스트셋의 오차 저장
y_vloss = history.history['val_loss']
y_loss = history.history['loss']

# 그래프 표현
x_len = np.arange(len(y_loss))
plt.plot(x_len, y_vloss, marker='.', c="red", label='Testset_loss') #=> 과적합 발생
plt.plot(x_len, y_loss, marker='.', c="blue", label='Trainset_loss')



plt.legend(loc='upper right')
plt.grid()
plt.xlabel('epoch')
plt.ylabel('loss')
plt.show()
```