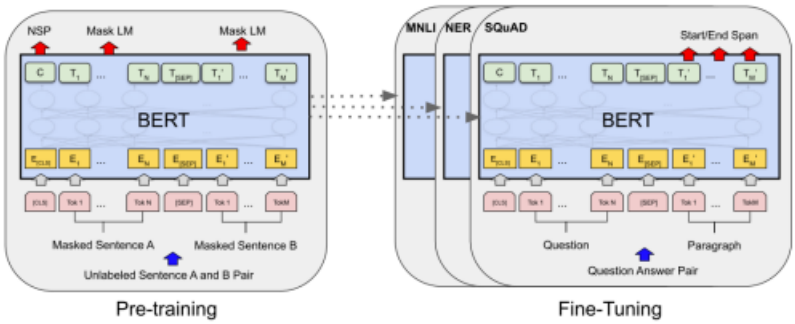
**BERT 모델

BERT 모델은 구글에서 발표된 사전 훈련 기반 딥러닝 언어모델이다. 이를 이용해 자연어 처리에서 사전 훈련을 적용할 수 있다. 즉 사전 훈련 모델로 새로운 모델을 만들어 다시 학습시키는 전이 학습이 가능하다. 이러한 전이학습은 이미 언어를 학습한 모델을 이용해 부가적인 것을 학습시키기 때문에 적은 데이터로 학습이 가능하다.

BERT 모델은 임베딩을 수행해 pre-trained된 BERT 위에 classification layer를 추가해 다양한 NLP를 처리한다(fine-tuning).



pretraining BERT: 언어를 이해하는 것

Fine tune BERT: 목적에 맞게 레이어를 추가하는 것

즉 pretraining으로 자연어를 토큰화하고 인코딩하여 기계가 이해할 수 있는 벡터 형태로 만들고, fine-tuning 단계에서 목적에 맞게 적절한 레이어를 추가 또는 파라미터를 조정하여 모델을 학습한다.

**학습 모델 레이어 분석

embedding: 사람이 쓰는 자연어를 기계가 이해할 수 있는 숫자형태인 vector로 바꾸는 과정이다.

conv1d: convolution(합성곱) 연산의 일부이다. conv1d, conv2d, conv3d가 나눠지는 기준은 합성곱이 진행되는 방향과 합성곱의 결과로 나오는 출력값이다. conv1의 합성곱의 방향과 출력 값은 아래와 같다.

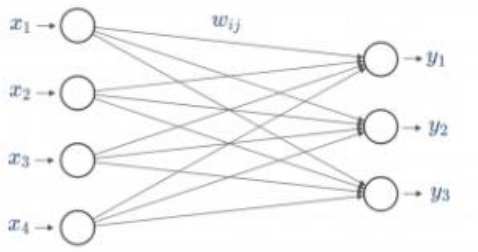
	합성곱의 방향	출력 값
conv1d	한 방향(가로)	1-D array(vector)

자연어 처리 분야에서 사용하는 합성곱의 경우 각 단어 벡터의 차원 전체에 대해 필터를 적용시키기 위해 주로 conv1d를 사용한다.

maxpool1d: pooling은 feature map의 크기를 줄이거나 주요한 특징을 뽑아 내기 위해 합성곱 신경망 이후에 적용하는 기법이다. 종류는 maxpool1d, maxpool2d, maxpool3d가 있다.

dropout: 과적합을 막기 위해 정규화 하는 방법으로 dropout 레이어를 추가한다. dropout은 신경망 기반 모형에 특화된 정규화 방법이다. 학습시킨 신경망 분류 결과는 epoch마다 비슷하지만 다른 여러 신경망 모델들의 결과를 평균내는 것과 유사하다. 이렇게 여러 개의 결과가 평균내기 때문에 일반적이지 않은 분류 결과를 제거하는 정규화 효과를 낸다.

dense: 신경망 구조의 가장 기본적인 형태로 y=f(Wx+b) 수식을 만족하는 기본적인 신경망 형태의 층을 만드는 레이어이다. (x=입력벡터, b=편향백터, W=가중치행렬, f=활성화함수)



신경망에서 활성화함수는 주로 시그모이드 함수 이용

시그모이드 함수: 시그모이드 함수를 이용하여 결과 값 y를 1또는 0에 맞춰 성공 또는 실패 두 가지로 분류한다. 아래는 시그모이드 함수의 식이다.

$$p(x) = \frac{1}{1 + e^{-(wx+b)}}$$

요즘에는 ReLU 함수를 활성화함수로 많이 이용

시그모이드 함수는 0보다 큰 값이면 1을 반환하는데 ReLU 함수는 0보다 작은 값이 나온 경우 0을 반환하고, 0보다 큰 값이 나온 경우 그 값을 그대로 반환한다. 즉 시그모이드 함수를 개선한 형태로 정확도가 더 좋다. 아래 그래프는 각각 sigmoid 함수와 ReLU 함수이다.

결국 hidden layer에는 ReLU를 적용하고 마지막 output layer에서만 시그모이드 함수를 적용하면

정확도가 올라간다.

output shape: (None, 5)는 None개의 행과 5개의 아웃풋 값이 주어졌다는 것을 의미한다. 행이 None로 지정되는 이유는 데이터의 개수는 계속해서 추가될 수 있기 때문에 딥러닝 모델에서는 주로 행을 무시하고 열의 shape를 맞추어 주는 작업을 수행한다.

param: 입력 노드와 출력 노드에 대해 연결된 간선의 수

**분석 리포트

[tf-idf]

tf: 특정 단어의 한 데이터 내 등장 빈도

idf:특정 단어의 여러 데이터 내 등장 빈도의 역.

=>tf-idf 어떠한 성질을 지니는지 알려줄 수 있는 그 단어를 추출하는 기법.

빈도수로 가치 정하는 것이 아니라 다른 데이터 내에서도 자주 등장하는 단어에 패널티(역수 값)를 줘서 스케일 맞춤.

이를 추출 후, train_test_split모듈로 train/test로 나눈 뒤, 이를 바탕으로 linearsvc(선형)실행.

tp(true positive) : true인거 true라고 예측/ fp(false positive):false인거 true라고 예측
tn(true negative) : false인거 false라고 예측/ fn(false negative):true인거 false라고 예측
정답=>tp, tn, 오답=>fp,fn

precision(정확도) => 결과가 얼마나 정확한 지 알려준다. tp/(tp+fp) true라고 분류한 것 중 진짜 true인 비율.

recall(재현율) => tp/(tp+fn). 실제 true인 것 중에서 true라고 예측한 비율.

f1-score => precision과 recall의 조화평균 = 2/((1/precision)+(1/recall))
(accuracy 지표를 보완한)성능을 평가하기 위한 좋은 지표

support => 표본의 개수(데이터 개수)

accuracy => tp+tn/(tp+fn+fp+tn), 전체 예측에서 true를 true로, false를 false라고 옳게 예측한 비율=> 가장 직관적으로 성능을 나타내는 지표.

macro avg => 평균들의 평균 개념. 평균의 합/평균 개수

weighted avg => 가중평균=(데이터 개수*수치)/개수 ex) 0.79*869+0.83*654/1523=0.81

**모델 학습

keras의 fit 함수를 통해 학습 과정을 표시.

epoch => 전체 데이터 학습할 횟수.

매 epoch마다 데이터를 훈련/검증 두 부분으로 분할. 훈련 데이터는 모델을 훈련시키고, 검증 데이터는 손실 및 정확도를 확인하는 검증을 한다. 그 결과로 loss(훈련 손실 값), accuracy(훈련 정확도), val_loss(검증 손실 값), val_accuracy(검증 정확도)나온다. val은 validation의 약자.

108 => 분할된 훈련 데이터 셋의 개수

loss는 결과 값과의 차이 => 0에 수렴할수록 성능 좋음.

accuracy 정확도 => 1에 수렴할수록 성능 좋음.

val_loss 증가, val_acc 감소 => 모델이 학습하지 않은 값을 입력

val_loss 증가, val_acc 증가 => 오버핏이 일어나거나 다른 현상의 가능성이 있다

val_loss 감소, val_acc 증가 => 학습이 제대로 작동.