Hyper parameter Tuning 대본

* **슬라이드 1(learning rate)**

다음은 Hyper parameter를 조정해본 결과입니다.

우선 Learning rate를 다양하게 수정해가며 결과를 봤는데요.

최초 best score를 달성했던 3epoh 기준으로 batch size를 128로 설정 했을 때 결과 입니다.

저희가 best score를 달성했던 5 -5승 learning rate에서는 최종 accuracy가 낮게 나왔고, 오히려 3 -5승 learning rate의 경우에 전체적인 accuracy면에서나 최종 accuracy가 높게 나왔습니다.

loss의 경우는 비슷한 수준으로 epoch마다 떨어지는 것을 보실 수 있습니다.

가장 결과가 좋았던 3 -5승 submission 결과는 0.984로 best score보다 다소 떨어지는 경향이 있었습니다..

* **슬라이드 2(epoch)**

다음은 epoch 수를 늘려서 loss를 최대한 떨어뜨려 보자라는 생각으로 5로 epoch를 늘려 learning rate 5개의 case에 대해 학습을 실행하였습니다.

.

생각대로 loss는 epoch마다 지속적으로 떨어지는 것을 확인할 수 있었고, accuracy또한 상승하는 것을 확인할 수 있었습니다.

accuracy 그래프를 보시면 초록색 선인 2 -5승 learning rate일때가 3 epoch 이후로 가장 높은 accuracy를 보이기 때문에, 해당 모델로 결과를 내어 submission을 해보았더니, score가 0.987로 best score를 달성했습니다.

* **슬라이드 3(hyper parameter tuning 결론)**

loss가 더 떨어질 수 있을까 궁금해서 epoch을 6으로 늘려보았을때, loss는 또 떨어지는 것을 볼 수 있었으나, accuracy는 오히려 낮아지므로 trainset에 대한 overfitting이 의심 되었고, epoch 수는 더 이상 늘리지 않는 것으로 결론을 지었습니다.

결과적으로, Test set에서 가장 좋은 성능을 보인 모델은 3 epoch, 5 -5승 일때 모델, 또한

5 epoch, 2 -5승 일때 모델 이었으며, epoch을 늘리는 것은 loss를 떨어뜨리는데 도움이 된다는 사실과 epoch이 클수록 learning rate는 작게 가져가는 것이 유리하다는 사실을

얻어 낼 수 있었습니다.

하지만, 새로운데이터, Test set에 대한 성능을 일정수준 이상으로 높이는데에는 한계가 있다는 것을 몸소 느끼게 되었고, 결국 일반화 수준을 높이는 방안, 즉 train 데이터의 개수를

늘려 보자는 아이디어로 다음 step에 접근을 해보았습니다.

* **슬라이드 4(Data switching idea)**

여러가지 방안을 고민해본 결과, 새로운 data를 가져와서 학습을 시키는 것은 부정행위의 소지가 다분하여.. 놀고있는 validation set의 데이터를 가져와 학습을 시켜보기로 생각을 하였습니다.

k-fold cross validation 등 다양한 기법이 있지만, 현 모델에 적용하기가 복잡하여, 간단하게 데이터를 epoch마다 바꿔볼까? 라는 생각을 하였고,

아래의 그림과 같이 epoch 마다 validation set의 일부, train set에 넣고 같은 수만큼의 train data를 validation set에 넣고 학습을 진행시켜보자라는 아이디어를 생각해보았습니다.

\*\*부연 설명: 그림과 달리 실제 데이터를 보면 train set이 validation set보다 훨씬 많기 때문에

아이디어를 표현을 하고자 Dataset의 도형을 같은 크기 표현한 것이고, 1 epoch이후 Validation set으로 들어간 Train data는 Train set으로 들어 Validation data와 **크기가 동일 합니다!**

**즉, 저 아이디어로 인해 1 epoch 이후 validation set의 크기가 점점 커지거나 하지 않음을 알려드립니다!**

* **슬라이드 5(Data switching code)**

따라서, 저희는 switch data 라는 함수를 작성하여 적용해보았고,

이 함수는 단순히 총 epochs를 기준으로 epoch마다 얼만큼의 데이터를 train set으로 넘길 것인지 결정하고,

베이스라인에서 train\_pos, train\_neg, dev\_pos, dev\_neg로 정의되었던, 리스트를 인수로 받아 train set과 validaiotn set의 데이터를 바꿔 주는 기능을 수행합니다.

급하게 작성한 코드이다보니 당연히 미흡한 부분도 있었는데요.

DataLoader로 들어간 데이터들을 바꿔주면 당연히 좋겠지만, 그 방법을 시도하다가 실패하여, 부득이하게 token id 값들만 가지고 있는 list를 기준으로 바꿔주었고, 따라서 epoch 마다 data를 다시 load해야합니다.

지금 저희가 가지고있는 데이터들로는 무리가 없었지만, heavy한 데이터에 적용했을 때, 이에따른 오버헤드가 점점 커질 것으로 생각이 됩니다.

또한, list를 인자로 하니, 원래의 list가 이 함수를 거치고 나면 바뀐상태로 있다보니, 기존의 아이디어대로 구현하려면, 아래 사진처럼 for문 안에서 2번함수가 호출되야 합니다.

이것 또한 비효율적인 코딩이라고 생각됩니다.

* **슬라이드 6(Data switching 결과)**

하지만, 위처럼 단점이 있음에도 불구하고, 결과는 나쁘지 않은 것으로 보였습니다.

우선 Accuracy는 일정 epoch이후로 큰 폭으로 상승함을 볼 수있었습니다.

다만, 생각을 좀더 해본 결과 학습을 했던 데이터가 validation set으로 들어가는 것이니 accuracy가 저렇게 높은 것은 당연하다고 판단했습니다.(괜히 기분은 좋습니다^//^)

loss의 경우도 여타 5 epoch을 돌린 모델과 비슷한 수치로 감소하였습니다.

이 모델로 나온 결과를 submission 했을 때의 score는 0.988으로 best score와 비교했을 때 0.001상승한 결과였습니다.

이 실험을 통해 얻은 결론은 다이나믹하게 높아진 결과는 아니지만, 데이터셋을 조금이라도 더 많이 가져왔을 때, 성능이 나아지긴 한다~ 이며,

그 외의 k-fold 같은 여타 cross validation 기술들도 충분히 적용해볼 가치가 있음을 알 수 있었습니다.