YUBERT

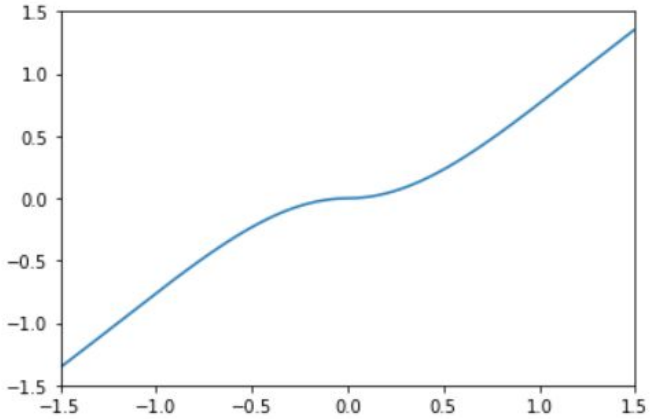
by beomgon.yu

기존 BERT의 문제점 및 개선 포인트 1

1. attention matrix(512*512)에서 softmax 시간 오래 소요
   -> fast transformer, linear 연산으로 바꿈(dot product , elu)
   -> vector의 내적으로 similarity를 구한 후 softmax를 취해 각 token별 가중치를 구하고 (합계 1) 가중합한다.
   1. 추가 고려사항, - 이미지에서 배경을 지우듯이 token별로 가중합할때 중요한 token에서 정보를 취하고 noise나 배경 같은 부분은 제거할 필요 있음
   (activation ft 수정), 관련 참고 논문(sparse transformer:concentrated attention)
   2. similarity를 구할 때 +뿐 아니라 -로 큰 값도 상관관계가 큰 값인데, softmax를 취하면 - 상관관계의 값이 모두 사라짐
   -> softmax대신 tanh를 취해서 pretrain/fine tune
   -> gradient vanishing 문제로 다른 activation or 위 1번 방법 사용해야 할듯

기존 BERT의 문제점 및 개선 포인트 1

New activation function,
ymish -> 0근처의 값은 0에 더 가깝게(background token등이 cls에 미치는 영향 최소화, +, -값은 그대로, vanishing gradient 해결
attention_probs = ymish(attention_scores/torch.norm(attention_scores, dim=3, keepdim=True) + 1e-9)




```
def swish(x):
    return x * torch.nn.Sigmoid()(x)

def ymish(x) :
    return x * torch.tanh(torch.abs(x))
```

기존 BERT의 문제점 및 개선 포인트 2

기존 bert에서는 token별 dimension이 768로 고정되어 있다.
iamge와 같이 feature extraction이 진행될 수록 size는 줄어들 고 dimension은 커질 수 없을까??

But NLP에서는 image와는 달리 token별 task도 많이 있고 해당 task를 수행함으로써 결과적으로 text에 대한 이해에도 도움이 될 수 있다. Bert 매우 단순하고 직관적임,

token별 dimension은 좀 줄여도 될 것이고, cls의 dimension(문단 전체를 포함하므로)은 늘려야 할 것이다.

how about more cls token??, 문단의 맨 처음에 cls token을 하나만 두지 말고 여러 개를 두자.
classifier 단에서 cls token들을 concat해주면 문단에 대한 vector의 dimension 커지는 효과 얻을 수 있다.(like image)
최소 앙상블의 효과로 인해 accuracy가 더 좋아질 것으로 예상됨.

기존 BERT의 문제점 및 개선 포인트 3

LinFormer,
multi head attention(ex 12)일 때, 512*512의 attention matrix가 총 12개 만들어진다. but rank는 그보다 훨씬 작을 것이다(재료 matrix가 512*64 * 64*512)

따라서 attention matrix를 만들 기 전에, linear projection을 한 번 더해서 seq_len을 줄이자는 아이디어, but linear projection을 key와 value에 각각 하게 되고, multi head attention * (총 layer개수)만큼 추가로 parameter가 들어가고, temporary memory도 추가로 더 필요할 것이다.

아예 multi head로 나누기 전에 먼저 linear projection을 하면 어떨까??
모든 token들이 동일하게 중요하지 않을 것이고 어떤 것은 어절이나 구절로 합쳐도 될 것이다. 압축의 효과가 있다.

But fixed size length의 조건이 필요하다.

기존 BERT의 문제점 및 개선 포인트 4

multi head attention을 사용함으로써 다양한 view에서 token간의 context를 뽑아내는 것이 가능하다.

일반적으로 단일 layer보다 그것을 여러 layer로 나누는 것이 feature들을 뽑아내고 학습하는데 더 유리하다.
(무작정 학습하는 것보다, constraints를 줌으로써 의도된 목적대로 학습을 강제할 수 있음, multi head attention, similarity를 이용 attention prob matrix를 학습하는 것이, 직접 attention prob matrix를 학습하는 것보다 더 좋음, synthesizer is not good in this point of view)

muti head attention이 꼭 필요하나 그 중 useless한 head도 존재
it means redundancy or ineffective training
각각의 head별로 서로 다른 activation을 주자, 각각의 head가 서로 다른 context를 보도록 constraints를 주는 효과
더 효율적으로 학습이 되지 않을까?
self.ativate_ft = [nn.Softmax(dim=-1), nn.SELU(inplace=False), nn.ELU(), nn.LeakyReLU(0.1, inplace=False), swish, ymish]
, 총 6개의 activation
split -> activate -> concat

Analyzing Multi-Head Self-Attention:Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned
Are Sixteen Heads Really Better than One?

리뷰 논문들

Transformers are RNNs:Fast Autoregressive Transformers with Linear Attention
https://arxiv.org/pdf/2006.16236.pdf

Are Sixteen Heads Really Better than One?
https://arxiv.org/pdf/1905.10650.pdf

Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel
https://arxiv.org/pdf/1908.11775.pdf

ON THE RELATIONSHIP BETWEEN SELF-ATTENTION AND CONVOLUTIONAL LAYERS
https://arxiv.org/pdf/1911.03584.pdf

Linformer: Self-Attention with Linear Complexity
https://arxiv.org/pdf/2006.04768.pdf

Multi-Head Attention:Collaborate Instead of Concatenate
https://arxiv.org/pdf/2006.16362.pdf

Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned
https://www.aclweb.org/anthology/P19-1580.pdf