Parameter_Efficient_Transfer_Learning_for_NLP

Overview

- 기존 transfer-learning 접근법은 parameter inefficient 했다.
  - 즉 task-specific한 많은 parameter를 추가해야 했다.
- 우리는 adaptor라는 compact model을 사용해서 parameter 추가를 최소화했다.
- BERT Transformer에 적용해서, 26개의 NLP task에 적용했는데, 기존 대비 3.6% parameter만 추가되고, 성능은 0.4% 정도만 감소

소개

모수 최소화가 필요한 상황

- Cloud 서비스에서 고객마다 다른 task가 실시간으로 풀어달라고 요청함 (in sequence)
- 이러한 online sequence of new task에 대해서 pre-trained 모델을 new task에 빠르게 adaptive하게 만들어야 한다.

Compact & Extensible Model 필요

- compact : 새로운 task에 대해 새로운 모수 추가를 최소화
- extensible : 점증적으로 새로운 task들에 적응하면서, 동시에 과거것을 잊어버리지 않아야 한다.

기존 전이학습 방법과의 차별점

- 기존 feature-based transfer
  - pretrained feature extractor 위에 new task마다 많은 모수를 써서 task-specific model 구성
- 기존 full-finetuning transfer
  - pretrained model 위에 small task-specific model을 구성하고, 학습시에는 기존 모수도 학습 대상
  - still inefficient in parameters
- 우리의 adaptor
  - pretrained model의 레이어 사이에 adaptor가 들어가고
  - bottleneck을 형성하고
  - 기존 모수들은 freeze해서 건드리지 않는다.
  - efficient in parameter

기타 학습방법과의 차이점

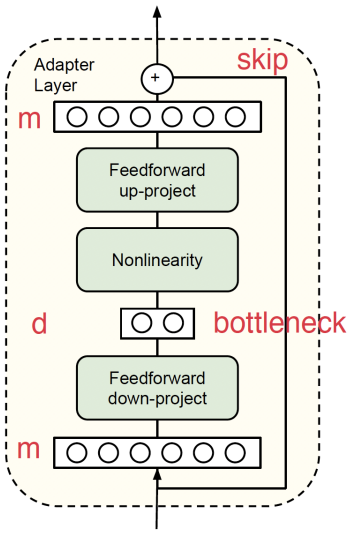
- Multi-task learning과의 차이점
  - 둘 다 많은 task를 대응한다는 결론은 같지만, 우리것은 한번에 하나의 task만 접근하면 되는 반면, multi-task learning은 동시에 모든 task에 접근해야 한다.
- Continual learning 과의 차이점
  - CL도 끊임없는 stream of task에 대응하는 것이 동일하지만, CL은 과거 것은 잊어버리는 경향이 있는 반면, 우리 것은 잊어어비리지 않는다.

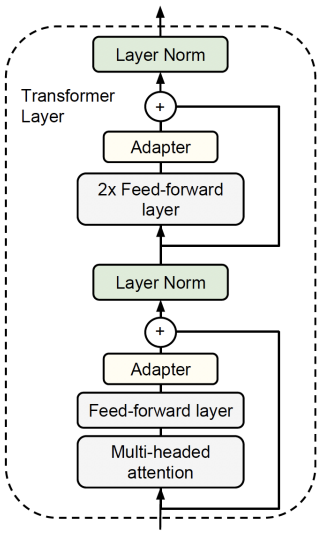
Adaptor-based tuning for text Transformer

- use 8% of the parameters of the original model

m << d

params
= 2md + d + m





Experiments

- Pretrained Transformer에서 나온 CLS 위에 linear layer 쌓아서 분류기

GLUE Benchmark

- full-fine-tuning은 9개의 task마다 기존 모수를 전체 학습이므로 9x
- Adaptor(8-256)은 task마다 최적의 adaptor size 선정
  - full-fine-tuning 대비 성능 감소는 최소화하는 대신 parameter efficient
- Adaptor 64은 task에 상관없이 고정 크기의 adaptor 선정
  - adaptor-size가 크게 민감하지는 않다.

Parameter-Efficient Transfer Learning for NLP

	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_m	MNLI_mm	QNLI	RTE	Total
BERT_LARGE	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

Additional (17)Tasks

- SOTA를 알 수 없어서 AutoML로 나름의 자체 SOTA 선정하고 비교
  - BERT를 쓰지는 않았음
- BERT + full-fine-tuning과 BERT + partial-fine-tuning과의 추가 비교

Dataset	No BERT baseline	$\text{BERT}_{\text{BASE}}$ Fine-tune	$\text{BERT}_{\text{BASE}}$ Variable FT	$\text{BERT}_{\text{BASE}}$ Adapters
20 newsgroups	91.1	92.8 ± 0.1	92.8 ± 0.1	91.7 ± 0.2
Crowdflower airline	84.5	83.6 ± 0.3	84.0 ± 0.1	84.5 ± 0.2
Crowdflower corporate messaging	91.9	92.5 ± 0.5	92.4 ± 0.6	92.9 ± 0.3
Crowdflower disasters	84.9	85.3 ± 0.4	85.3 ± 0.4	84.1 ± 0.2
Crowdflower economic news relevance	81.1	82.1 ± 0.0	78.9 ± 2.8	82.5 ± 0.3
Crowdflower emotion	36.3	38.4 ± 0.1	37.6 ± 0.2	38.7 ± 0.1
Crowdflower global warming	82.7	84.2 ± 0.4	81.9 ± 0.2	82.7 ± 0.3
Crowdflower political audience	80.8	80.9 ± 0.3	80.7 ± 0.8	79.0 ± 0.5
Crowdflower political bias	76.8	75.2 ± 0.9	76.5 ± 0.4	75.9 ± 0.3
Crowdflower political message	43.8	38.9 ± 0.6	44.9 ± 0.6	44.1 ± 0.2
Crowdflower primary emotions	33.5	36.9 ± 1.6	38.2 ± 1.0	33.9 ± 1.4
Crowdflower progressive opinion	70.6	71.6 ± 0.5	75.9 ± 1.3	71.7 ± 1.1
Crowdflower progressive stance	54.3	63.8 ± 1.0	61.5 ± 1.3	60.6 ± 1.4
Crowdflower US economic performance	75.6	75.3 ± 0.1	76.5 ± 0.4	77.3 ± 0.1
Customer complaint database	54.5	55.9 ± 0.1	56.4 ± 0.1	55.4 ± 0.1
News aggregator dataset	95.2	96.3 ± 0.0	96.5 ± 0.0	96.2 ± 0.0
SMS spam collection	98.5	99.3 ± 0.2	99.3 ± 0.2	95.1 ± 2.2
Average	72.7	73.7	74.0	73.3
Total number of params	—	17×	9.9×	1.19×
Trained params/task	—	100%	52.9%	1.14%

Performance/Parameter trade-off

- adaptor size를 줄이면, 더 efficient하지만 성능 감소가 없지는 않다.
- 하지만 fine-tuning에 비하면 trade-off 관계가 강하지는 않다.
- 아래는 task들의 평균 도식

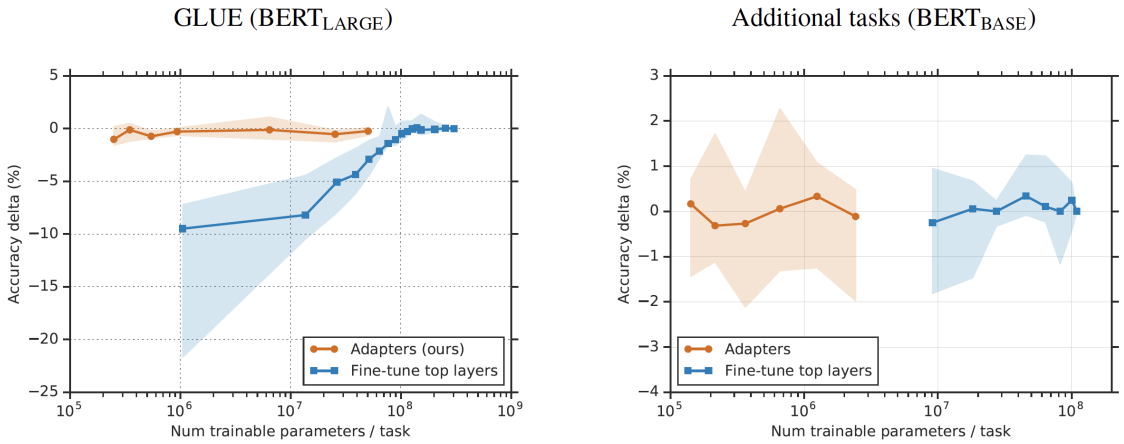
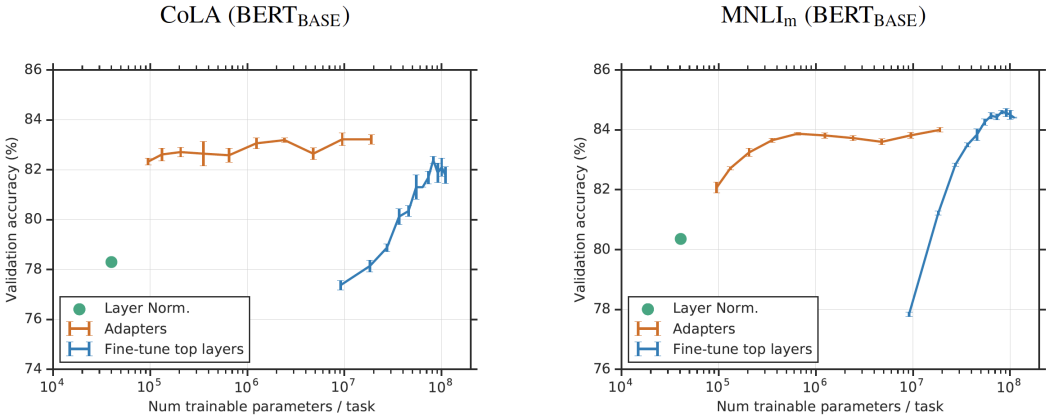


Figure 3. Accuracy versus the number of trained parameters, aggregated across tasks. We compare adapters of different sizes (orange) with fine-tuning the top n layers, for varying n (blue). The lines and shaded areas indicate the 20th, 50th, and 80th percentiles across

- 아래는 특정 두 개의 task에 대한 도식
  - LayerNorm은 Transformer의 Normalizer Layer만 학습 대상으로 삼은 것



Analysis and Discussion

전반적으로 higher layer의 adaptor가 더 중요한 역할

- 아래처럼 하나씩 adaptor를 제거했을 때의 성능 감소 테스트

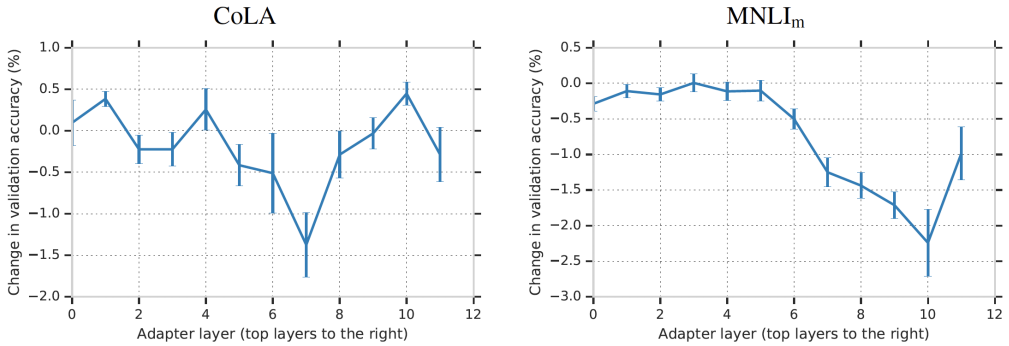


Figure 5. Ablation of trained adapters from each layer of BERT. *x-axis*: The index of the layer whose adapters are removed. *y-axis*: Relative performance of the model before and after ablation. Values smaller than zero indicate a decrease in performance after ablation. The line indicates the mean across three models trained with different random seeds, and the error bars indicate ± 1 s.e.m.

Adator의 weight 초기값은 너무 크면 안된다. << 0.01

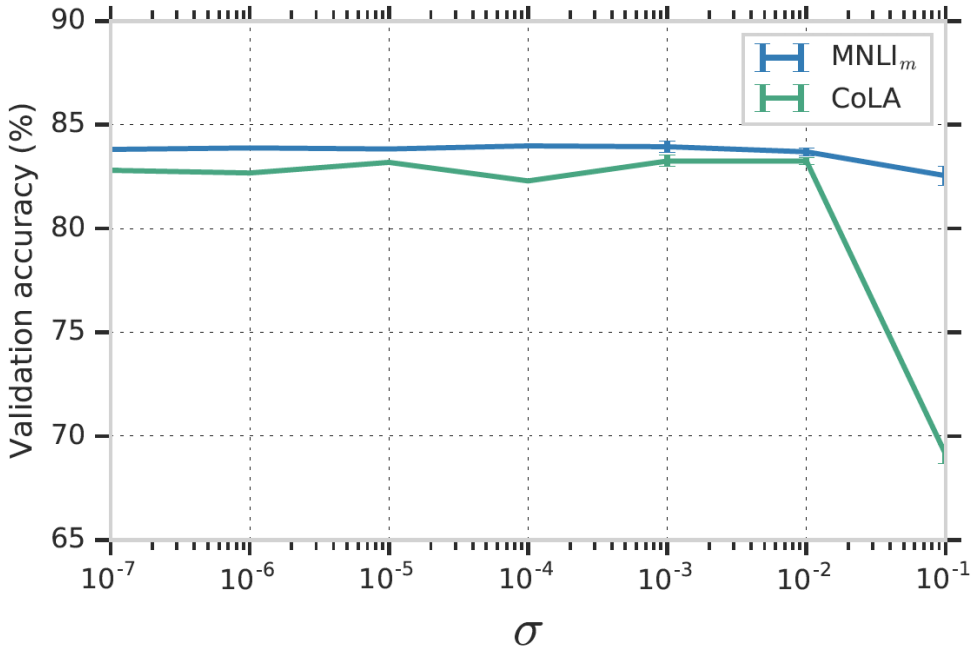


Figure 6. Performance of the BERT$_{\text{BASE}}$ model with adapters under varying weight magnitude at initialization. The weights are

Adator의 변형 실험들을 했으나 Bottleneck 구조 대비 별 향상 없다.

- add BatchNorm to adator
- adator 내의 레이어 더 늘리기
- different activation functiomn
- insert adaptor only to the attention layer of Transformer
- Adding adaptor parrallel to the main layers

언급이 없어서 아쉬운 점

task가 sequential하게 올 때, 어떻게 하는 것인지에 대한 언급이 없음

- task 추가시마다 new adaptor가 더 append되는 것인지 아닌지, 그냥 기존 adaptor만 계속 학습하면 되는 것인지

forgetting이 안 일어나는지에 대한 실험이나 상세 설명이 없음

In [ ]: