DistilBERT

A Distilled Version of BERT

Index

- Introduction
- Knowledge Distillation
  - Cross Entropy Temperature
  - Dark Knowledge
  - Learning Process
- DistilBERT
  - Architecture
  - Objective Function
  - Performance
- Conclusion

Introduction

BERT는 규모가 크다…

거대한 규모에서 발생하는 문제

- 여러 개의 query에 대해 빠르게 결과를 받아야 할 때??
- 용량이 작은 device에서 사용되어야 할 때??

Parameter quantization, pruning, distillation 중 distillation에 초점을 맞춘 DistilBER

Knowledge Distillation

Teacher-Student Learning

[사전적 정의]

- Distillation(증류): 온도차(끓는점의 차이)를 이용해서 혼합물의 성분을 분리하는 방법

모델 압축 기술

- **사전에 학습된 거대한 모델**(Teacher Network)에서 얻은 **지식**을 **작은 모델**(Student Network)에게 학습시키는 것

Knowledge Distillation

Cross Entropy Temperature

Cross Entropy Temperature

확률 분포 평탄화 -> 다른 레이블의 확률값을 높일 수 있음 (near-zero의 값 증대)

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_i}{T}\right)}$$

T	Label1	Label2	Label3
1	0.997	0.002	0.001
2	0.935	0.046	0.017
5	0.637	0.191	0.021

Knowledge Distillation

Dark Knowledge

잘 훈련된 모델에서는 정답 label을 제외한 나머지 label의 확률값이 거의 0에 가깝다 -> near zero

Temperature에 의해 평탄화된 확률분포 -> 모델의 generalization에 기여하는 부분은 평탄화 되기 전의 near-zero

분류문제를 가정하고 얻어진 softmax 값에서 argmax label을 제외한 다른 출력값이 어느정도 높다면 다른 label에 비해 상대적으로 정답과 연관된 label일 수 있다 -> 정답 label의 확률 뿐만 아니라 다른 label의 확률도 학습에 사용하자.

회귀 모델에 적용된 것: https://arxiv.org/pdf/2002.12597.pdf

Knowledge Distillation

Learning Process

Soft Labels(targets), Soft Predictions

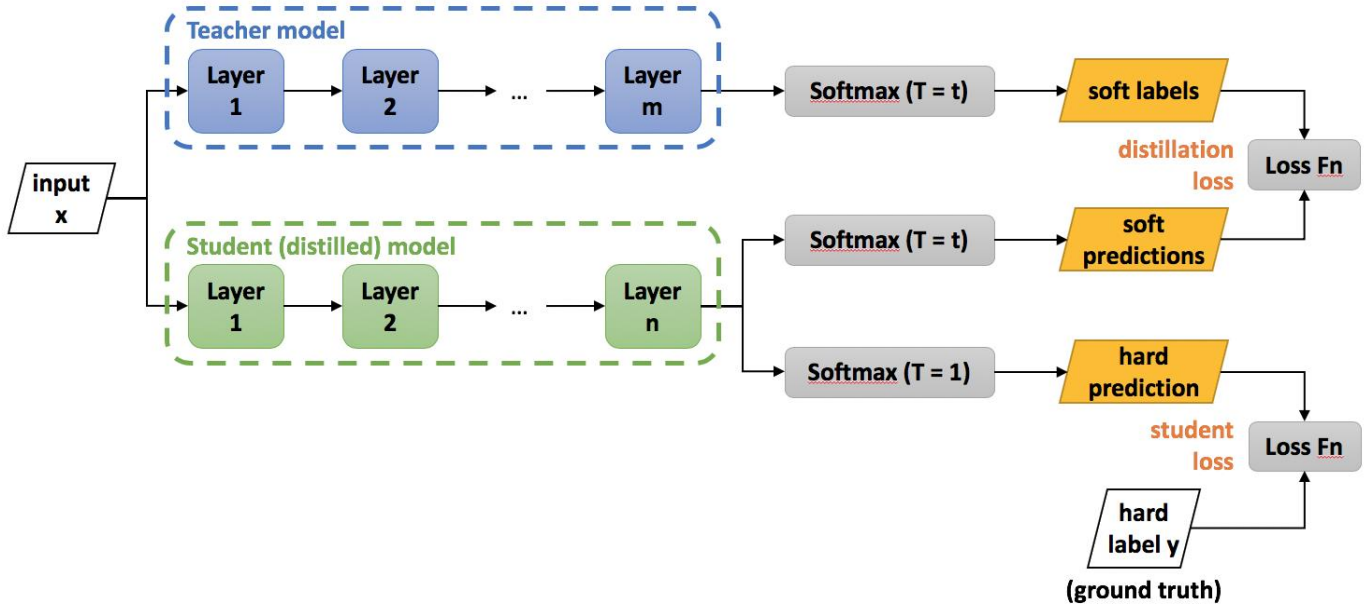
- T>1 일때의 cross entropy temperature distribution

Hard prediction

- T=1 student cross entropy temperature

Hard Labels(targets)

- teacher network의 ground truth
- argmax label의 값은 1로, 나머지는 0으로



Objective Function = Minimize (alpha*Distillation Loss + beta*Student Loss)

DistilBERT

Architecture

DistilBERT와 BERT-based의 Architecture차이

- token type embedding layer 제거
- pooler 제거
- layer 수를 2배 감소

Dimmension은 동일하기 때문에 teacher model의 weight을 그대로 사용

DistilBERT

Objective Function

LOSS = MLM Loss (학생 손실 = teacher model의 ground truth의 차이)

+ Teacher Model과 Student Model의 distribution의 차이(증류 손실)

+ Cosine Embedding LOSS

$$loss = \omega_1 \cdot loss_{MLM} + \omega_2 \cdot loss_{kL} + w_3 \cdot loss_{\cos}$$

DistilBERT

Performance

Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

BERT-based 대비 97%의 성능, 40% 적은 parameter

Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

BERT-based보다 40~60% 빠름

Conclusion

DistilBERT는 기존에 학습된 BERT-base model을 distillation을 통해 작은 규모의 네트워크로 학습시켜 성능은 약간 감소하였지만, 더 빠른 연산 속도와 더 적은 용량의 효율적인 네트워크를 만들 수 있었다.