지식 증류를 이용한 한국어 RoBERTa의 경량화

Knowledge distillation for lightweight RoBERTa of Korean

강동찬[01], 나승훈[1], 최윤수[2], 이혜우[2], 장두성[2]

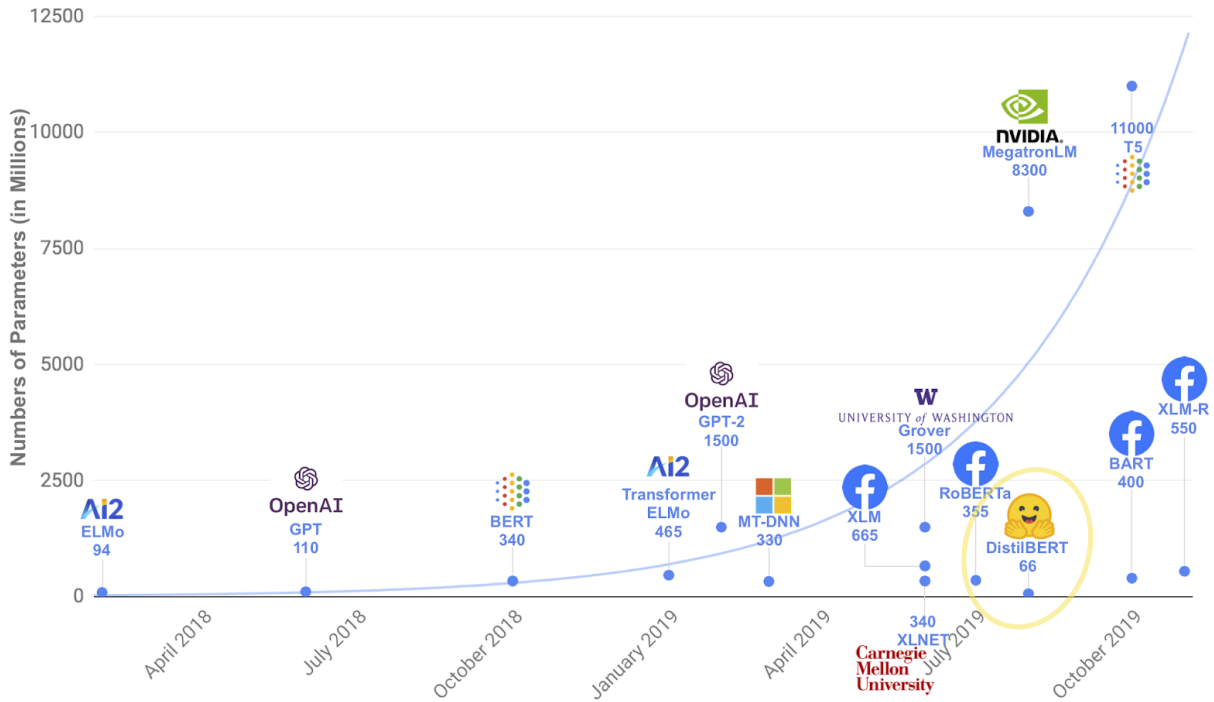
[1]전북대학교, [2]KT

Contents

Introduction: trends and problems

Trends

- 최근 NLP 분야에서 BERT와 같이 사전학습 된 언어모델들은 state-of-the-art 인코더
- 사전학습 된 언어모델의 크기는 점점 커지고 있는 추세



Problems

1. **Memory usage**
   - 단말기에 탑재하기 어려움
   - 확장 되기 어려움

2. **Inference speed**
   - 서비스 되기 어려운 속도

3. **Environmental cost**
   - 연구에 사용되는 환경적인 비용

Problems

1. **Memory usage**
   - 단말기에 탑재하기 어려움
   - 확장 되기 어려움

2. **Inference speed**
   - 서비스 되기 어려운 속도

3. **Environmental cost**
   - 연구에 사용되는 환경적인 비용

Financial cost

연구/개발 단계에서 재정적인 비용과 직접적으로 연관

Problems

1. **Memory usage**
   - 단말기에 탑재하기 어려움
   - 확장 되기 어려움

2. **Inference speed**
   - 서비스 되기 어려운 속도

3. **Environmental cost**
   - 연구에 사용되는 환경적인 비용

Consumption	**CO_2e (lbs)**
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]

Energy and Policy Considerations for Deep Learning in NLP [Strubell et al., ACL 2019]

Related work: DistilBERT and TinyBERT

DistilBERT

- **지식 증류 (Knowledge Distillation)**
  큰 교사 모델의 아웃풋 분포를 작은 모델의 학습에 이용하여, 작은 학생 모델의 학습을 도움

$$t^i = \frac{\exp(z_i^{\mathcal{T}}/T)}{\sum_j \exp(z_i^{\mathcal{T}}/T)}$$

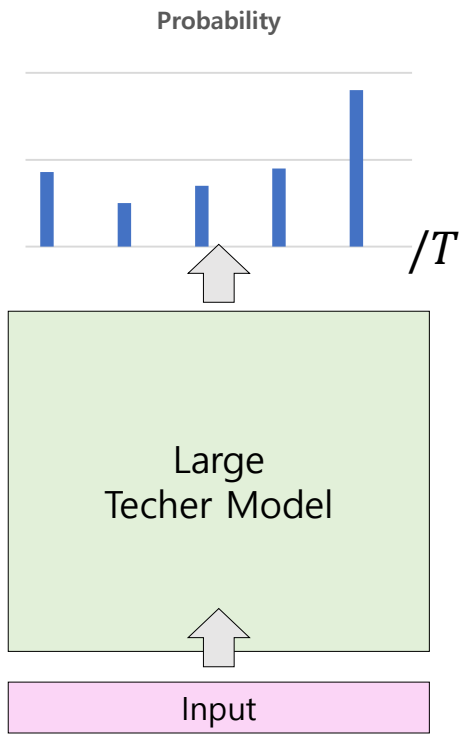
$$s^i = \frac{\exp(z_i^{\mathcal{S}}/T)}{\sum_j \exp(z_j^{\mathcal{S}}/T)}$$

$$\mathcal{L}_{kd} = -\sum_i t^i * \log(s^i)$$

Distilling the knowledge in a neural network. [Hinton et al., NIPS workshop, 2015]

DistilBERT

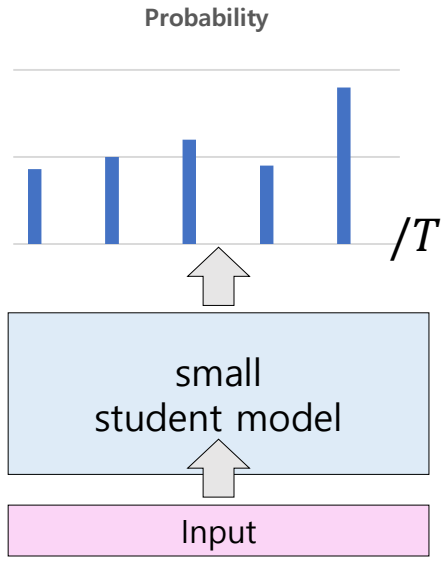
- **지식 증류 (Knowledge Distillation)**



$$t^i = \frac{\exp(z_i^{\mathcal{T}}/T)}{\sum_j \exp(z_i^{\mathcal{T}}/T)}$$

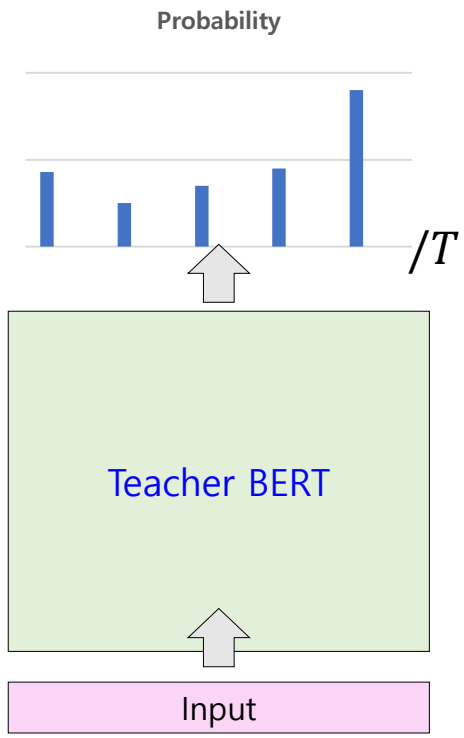
$$s^i = \frac{\exp(z_i^{\mathcal{S}}/T)}{\sum_j \exp(z_j^{\mathcal{S}}/T)}$$

$$\mathcal{L}_{kd} = -\sum_i t^i * \log(s^i)$$



DistilBERT

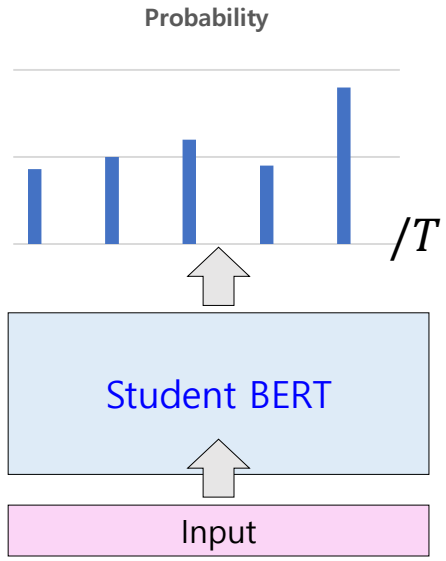
- **Key idea**



$$t^i = \frac{\exp(z_i^{\mathcal{T}}/T)}{\sum_j \exp(z_i^{\mathcal{T}}/T)}$$

$$s^i = \frac{\exp(z_i^{\mathcal{S}}/T)}{\sum_j \exp(z_j^{\mathcal{S}}/T)}$$

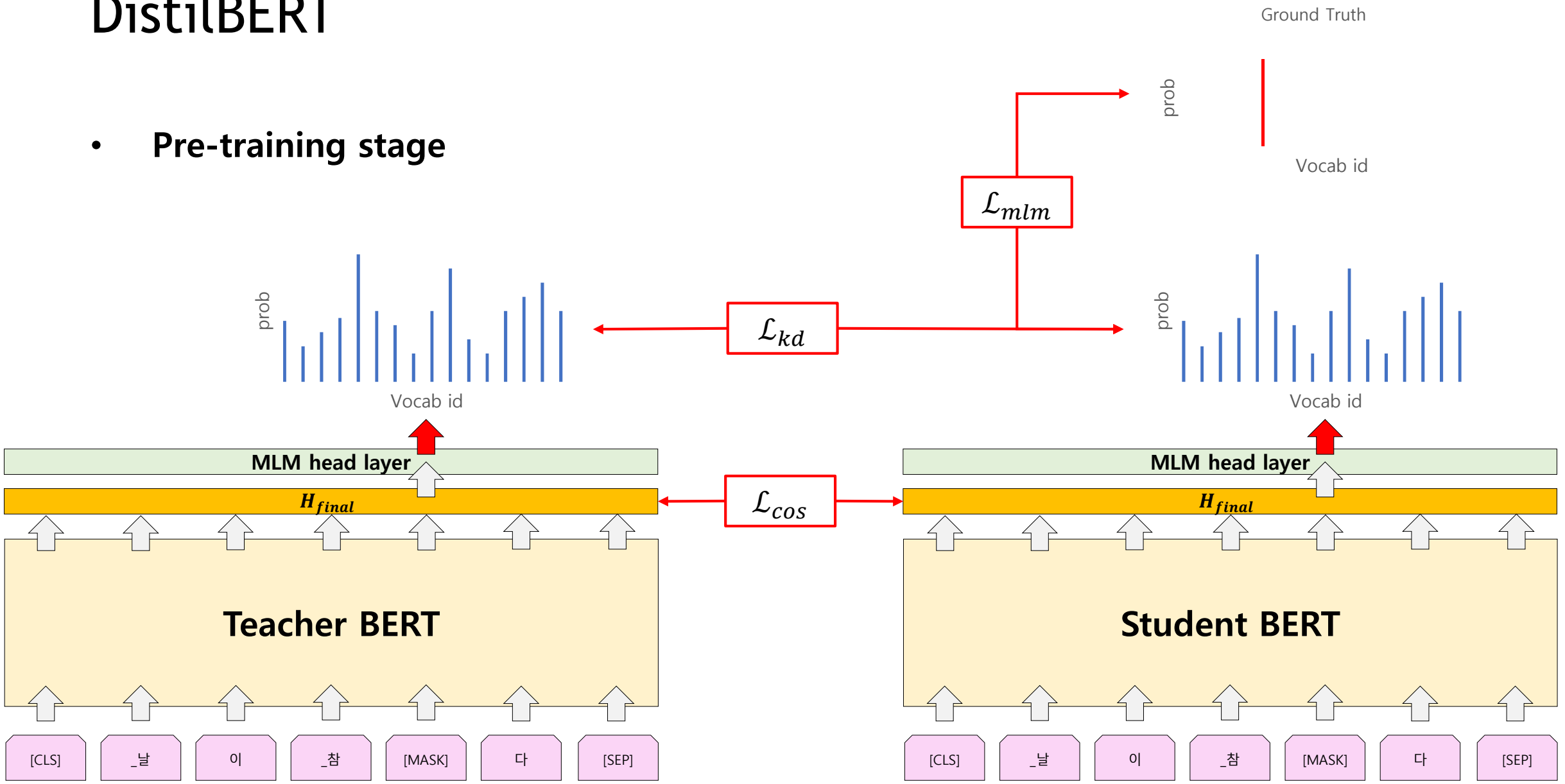
$$\mathcal{L}_{kd} = -\sum_i t^i * \log(s^i)$$



$$\mathcal{L} = \alpha_{mlm}\mathcal{L}_{mlm} + \alpha_{kd}\mathcal{L}_{kd} + \alpha_{cos}\mathcal{L}_{cos}$$

DistilBERT

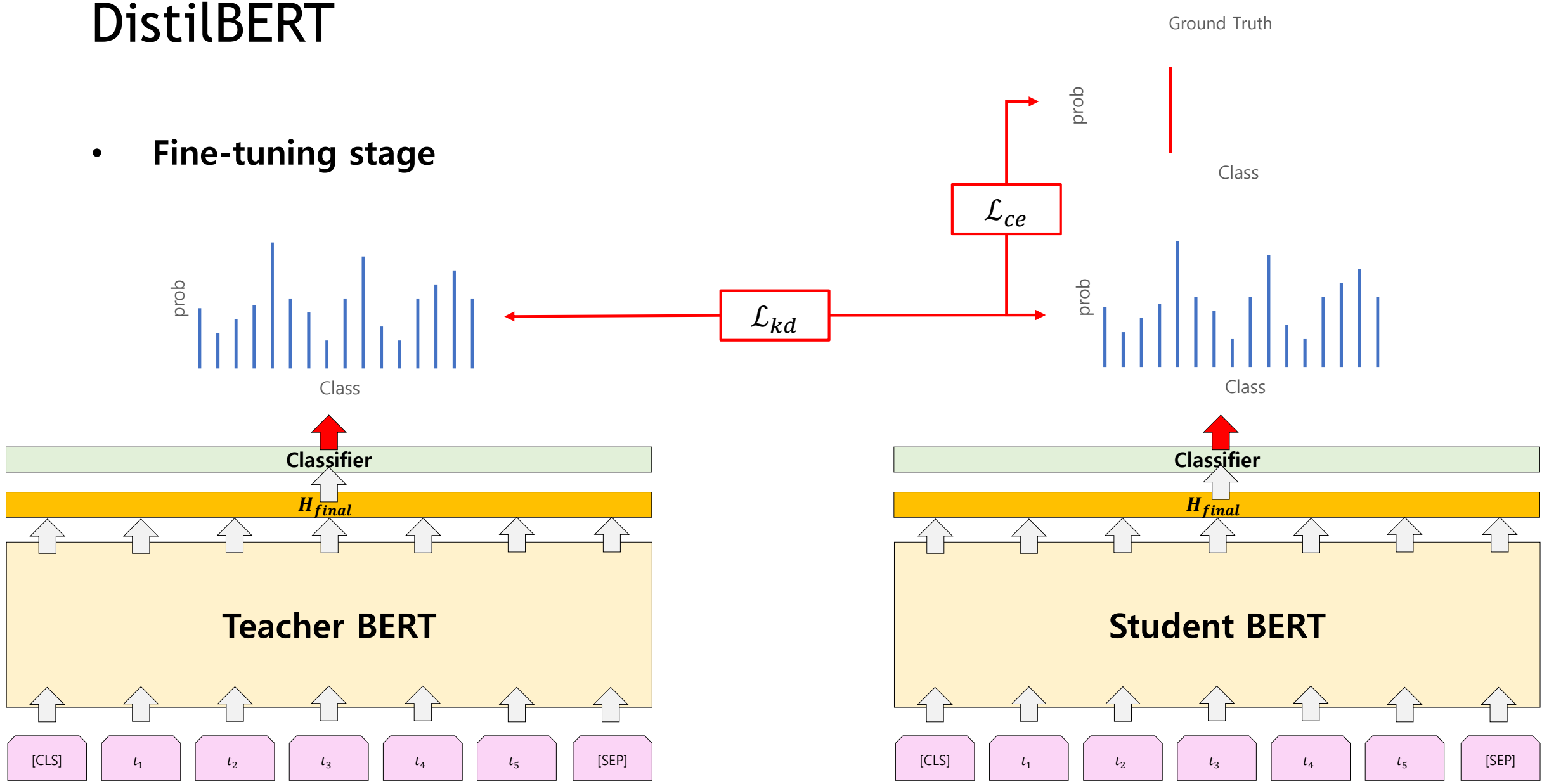
- **Pre-training stage**



$$\mathcal{L} = \alpha_{ce}\mathcal{L}_{ce} + \alpha_{kd}\mathcal{L}_{kd}$$

DistilBERT

- **Fine-tuning stage**



TinyBERT

- **트랜스포머 증류 (Transformer Distillation)**

  TinyBERT에서 제안한 교사 BERT모델의 인코더들의 아웃풋 표현을 이용한 증류법

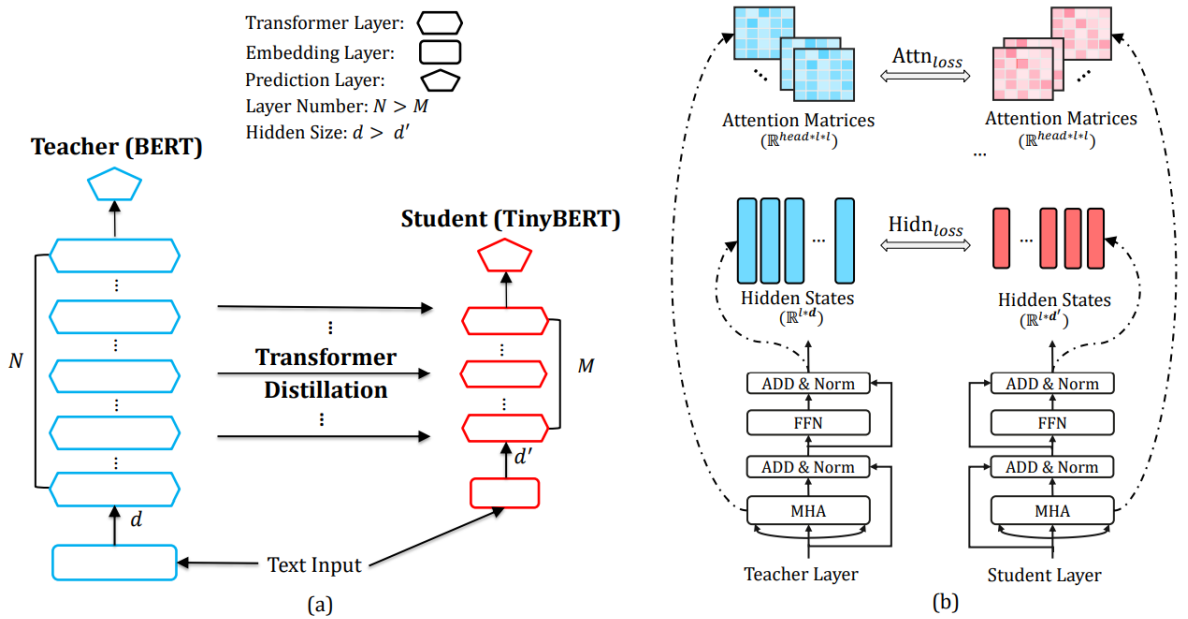
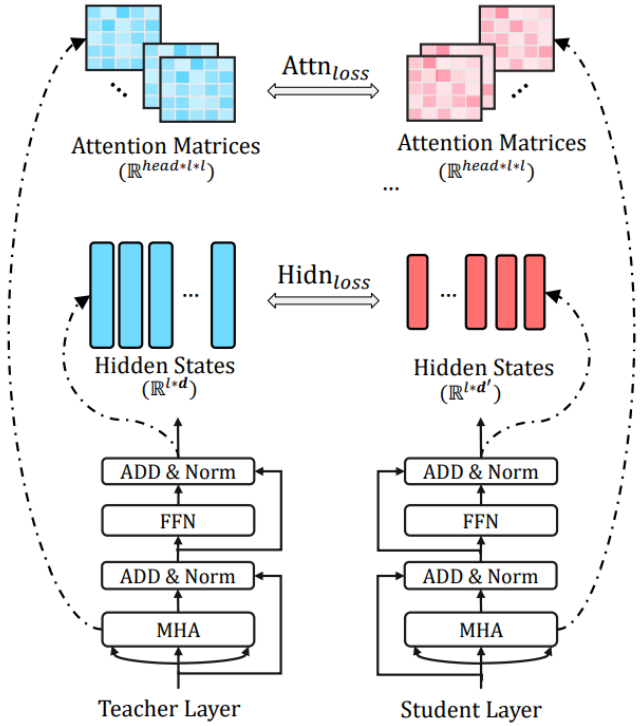


Figure 1: An overview of Transformer distillation: (a) the framework of Transformer distillation, (b) the details of Transformer-layer distillation consisting of Attn$_{loss}$(attention based distillation) and Hidn$_{loss}$(hidden states based distillation).

TinyBERT: Distilling BERT for Natural Language Understanding [Jiao et al., Arxiv 2019]

TinyBERT

- **Key idea**



$$\mathcal{L}_{\text{embd}} = \text{MSE}\left(E^{\mathcal{S}}W_e, E^{\mathcal{T}}\right)$$

$$\mathcal{L}_{\text{hidn}} = \text{MSE}\left(H^{\mathcal{S}}W_h, H^{\mathcal{T}}\right)$$

$$\mathcal{L}_{\text{attn}} = \frac{1}{h}\sum_{i=1}^{h}\text{MSE}\left(A_i^{\mathcal{S}}, A_i^{\mathcal{T}}\right)$$

$$\mathcal{L}_{\text{pred}} = -\sum_i t^i * \log\left(s^i\right)$$

$$\mathcal{L}_{\text{layer}}\left(S_m, T_{g(m)}\right) = \begin{cases} \mathcal{L}_{\text{embd}}(S_0, T_0), & m = 0 \\ \mathcal{L}_{\text{hidn}}\left(S_m, T_{g(m)}\right) + \mathcal{L}_{\text{attn}}\left(S_m, T_{g(m)}\right), & M \geq m > 0 \\ \mathcal{L}_{\text{pred}}(S_M, T_{N+1}), & m = M+1 \end{cases}$$

$$\mathcal{L} = \sum_{m=0}^{M+1}\lambda_m \mathcal{L}_{\text{layer}}\left(S_m, T_{g(m)}\right)$$

TinyBERT

- **Pre-training stage, Fine-tuning stage**

KD Methods	KD at Pre-training Stage					KD at Fine-tuning Stage				
	INIT	Embd	Attn	Hidn	Pred	Embd	Attn	Hidn	Pred	DA
Distilled $BiLSTM_{SOFT}$									✓	✓
BERT-PKD	✓							✓[3]	✓	
DistilBERT	✓				✓[4]				✓	
TinyBERT (our method)		✓	✓	✓		✓	✓	✓	✓	✓

Our Work: Two stage Distillation in fine-tuning stage

Motivation

- **Stage-wise training in Fitnets**

  교사 모델로부터 표현 학습 후, KD 적용.

Algorithm 1 FitNet Stage-Wise Training.
The algorithm receives as input the trained parameters $\mathbf{W_T}$ of a teacher, the randomly initialized parameters $\mathbf{W_S}$ of a FitNet, and two indices h and g corresponding to hint/guided layers, respectively. Let $\mathbf{W_{Hint}}$ be the teacher's parameters up to the hint layer h. Let $\mathbf{W_{Guided}}$ be the FitNet's parameters up to the guided layer g. Let $\mathbf{W_r}$ be the regressor's parameters. The first stage consists in pre-training the student network up to the guided layer, based on the prediction error of the teacher's hint layer (line 4). The second stage is a KD training of the whole network (line 6).

Input: $\mathbf{W_S}, \mathbf{W_T}, g, h$
Output: $\mathbf{W_S^*}$
1: $\mathbf{W_{Hint}} \leftarrow \{\mathbf{W_T}^1, \ldots, \mathbf{W_T}^h\}$
2: $\mathbf{W_{Guided}} \leftarrow \{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\}$
3: Intialize $\mathbf{W_r}$ to small random values
4: $\mathbf{W^*_{Guided}} \leftarrow \underset{\mathbf{W_{Guided}}}{\operatorname{argmin}}\ \mathcal{L}_{HT}(\mathbf{W_{Guided}}, \mathbf{W_r})$
5: $\{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\} \leftarrow \{\mathbf{W_{Guided}}^{*1}, \ldots, \mathbf{W_{Guided}}^{*g}\}$
6: $\mathbf{W_S^*} \leftarrow \underset{\mathbf{W_S}}{\operatorname{argmin}}\, \mathcal{L}_{KD}(\mathbf{W_S})$

Fitnets: Hints for thin deep nets [Romero et al., ICLR, 2015]

Our Work

- **Inspired by stage-wise training in Fitnets**

  교사 BERT로부터 표현 학습($\mathcal{L}_{trm}$) 후, KD 학습($\mathcal{L}_{pred}$). 각 파인튜닝 데이터 셋의 Train set 의 10%를 Validation set 으로 사용

$$\mathcal{L}_{\text{layer}}\left(S_m, T_{g(m)}\right) = \begin{cases} \mathcal{L}_{\text{embd}}(S_0, T_0), & m = 0 \\ \mathcal{L}_{\text{hidn}}\left(S_m, T_{g(m)}\right) + \mathcal{L}_{\text{attn}}\left(S_m, T_{g(m)}\right), & M \geq m > 0 \end{cases}$$

$$\mathcal{L}_{\text{trm}} = \sum_{m=0}^{M} \lambda_m \mathcal{L}_{\text{layer}}\left(S_m, T_{g(m)}\right)$$

$$\mathcal{L}_{\text{pred}} = f\left(Z^{\mathcal{S}}, Z^{\mathcal{T}}\right)$$

Experiment

Experiment setup (pre-training stage)

- **Corpus**

한국어 위키피디아 + 뉴스 + 백과사전 = 15GB

- **Configuration**

	Hidden	Embd	Intermediate	#Attention heads	#Hidden layers	Vocab	#params
Teacher RoBERTa	768	768	3072	12	12	31331	110M
DistilRoBERTa	768	768	3072	12	6	31331	66M
TinyRoBERTa	312	312	1200	12	4	31331	14M

Experiment setup (fine-tuning stage)

- **Dataset**

NSMC, KorQuAD, ETRI NER

- **KD function**

NSMC	$\mathcal{L}_{\text{pred}} = MSE(Z^{\mathcal{S}}, Z^{\mathcal{T}})$
NER	$\mathcal{L}_{\text{pred}} = MSE(Z^{\mathcal{S}}, Z^{\mathcal{T}})$
KorQuAD	$\mathcal{L}_{\text{pred}} = \{MSE(Z^{\mathcal{S}}_{\text{start}}, Z^{\mathcal{T}}_{\text{start}}) + MSE(Z^{\mathcal{S}}_{\text{end}}, Z^{\mathcal{T}}_{\text{end}})\}/2$

Experiment result

- **Result**

	corpus size	#params	NSMC (Acc)	KorQuAD 1.0 (EM / F1)	ETRI NER (F1)
BERT-Multilingual [8, 9]	-	110M	87.43	77.69 / 89.98	- / 91.92
BERT-ETRI [9]	23.5G	110M	-	84.72 / 92.74	-
BERT-형태소 태그 [8]	540M	110M	86.57	-	- / 91.58
RoBERTa-형태소 태그 [10]	15G	110M	89.88	-	- / 94.79
RoBERTa-자소 (Teacher) [11]	18G	110M	91.14	87.11 / 94.47	92.80 / 93.47
DistilRoBERTa	15G	66M	89.93	82.99 / 91.42	92.13 / 91.75
+ pred			**90.02**	**83.97 / 92.47**	92.22 / 92.09
+ trm + pred			89.89	83.13 / 91.66	90.73 / **92.31**
TinyRoBERTa	15G	14M	87.20	72.01 / 83.22	86.88 / **85.70**
+ pred			87.87	75.16 / 86.31	87.15 / 84.58
+ trm + pred			**88.20**	**75.90 / 87.05**	87.69 / 85.05

Conclusion and future work

Conclusion and future work

- **Conclusion**

  본 연구에서는 DistilBERT와 TinyBERT의 방법론을 사전학습 단계에 적용하여 기존 연구와 비교하고, 파인튜닝 단계에서 2단계 증류를 통해 효과를 확인하였다. 실험결과 파인튜닝 단계에서 예측 레이어의 증류는 성능의 향상에 도움이 되었던 한편, 트랜스포머 증류의 경우는 예측 레이어의 성능을 저하시키는 요인으로 작용하기도 하였다.

- **Future work**

  추후연구에서는 더 다양한 경량화 방법을 시도하고, 교사 모델의 성능과 차이가 많이 나는 태스크에서 성능차이를 줄이고자 한다.

Thanks for your attention !