

05

Training

영어 - 독일어

- 약 450만 문장 쌍
- 바이트-쌍 인코딩(BPE) 사용
- 약 37,000개 어휘 토큰

영어 - 프랑스어

- 약 3,600만 문장 쌍
- 워드피스 토큰화
- 약 32,000개 어휘 토큰

배치 방식: 문장 쌍을 시퀀스 길이에 맞게 배치, 각 배치에 소스와 타겟 토큰 각각 약 25,000개 포함

영어 - 독일어

- 약 450만 문장 쌍
- 바이트-쌍 인코딩(BPE) 사용
- 약 37,000개 어휘 토큰

영어 - 프랑스어

- 약 3,600만 문장 쌍
- 워드피스 토큰화
- 약 32,000개 어휘 토큰

배치 방식: 문장 쌍을 시퀀스 길이에 맞게 배치, 각 배치에 소스와 타겟 토큰 각각 약 25,000개 포함

5.2 Hardware and Schedule

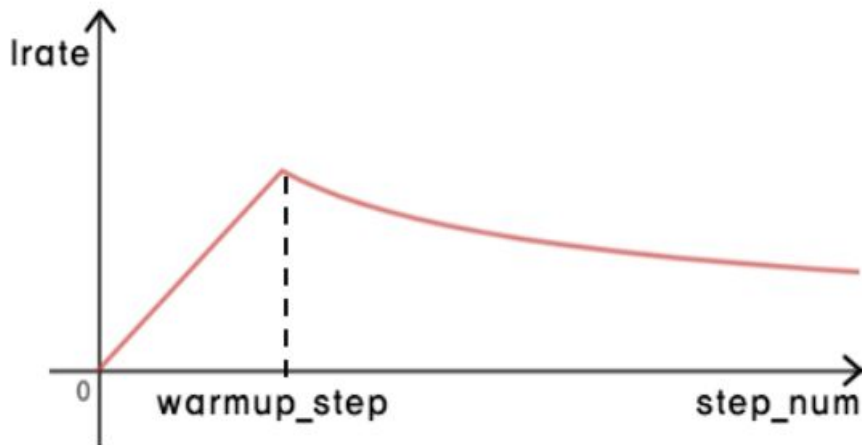
- **하드웨어:** 8개의 NVIDIA P100 GPU가 장착된 하나의 머신
- **훈련 시간:**
 - ★ Base 모델: 훈련 100,000단계 (12시간)
 - ★ Big 모델: 훈련 300,000단계 (3.5일)
- **훈련 속도:** Base 모델의 각 훈련 단계는 약 0.4초, Big 모델은 1.0초 소요

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5})$$

Adam

Learning rate scheduler(학습률 스케줄링): 학습률을 동적으로 변화

- 훈련 초기에는 학습률을 선형적으로 증가시키고, 이후에는 **step number**(단계 수)의 **inverse square root**(역제곱근)에 비례하여 감소
- `warmup_steps = 4000`

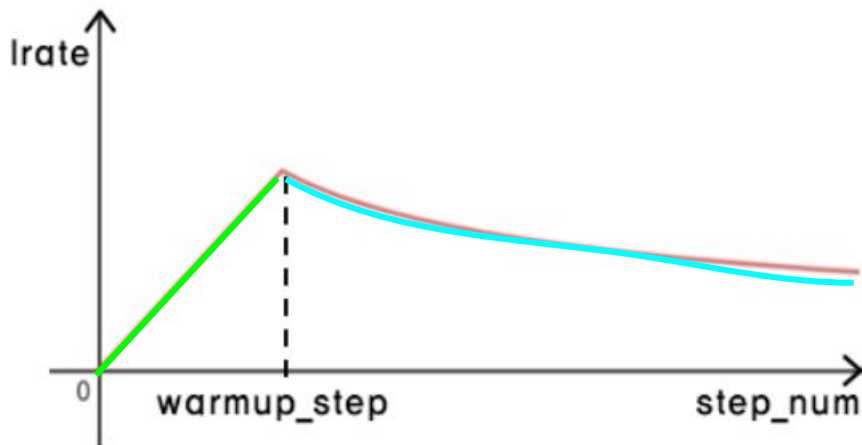


$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5})$$

Adam

Learning rate scheduler(학습률 스케줄링): 학습률을 동적으로 변화

- 훈련 초기에는 학습률을 선형적으로 증가시키고, 이후에는 step number(단계 수)의 inverse square root(역제곱근)에 비례하여 감소
- warmup_steps = 4000



$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5})$$

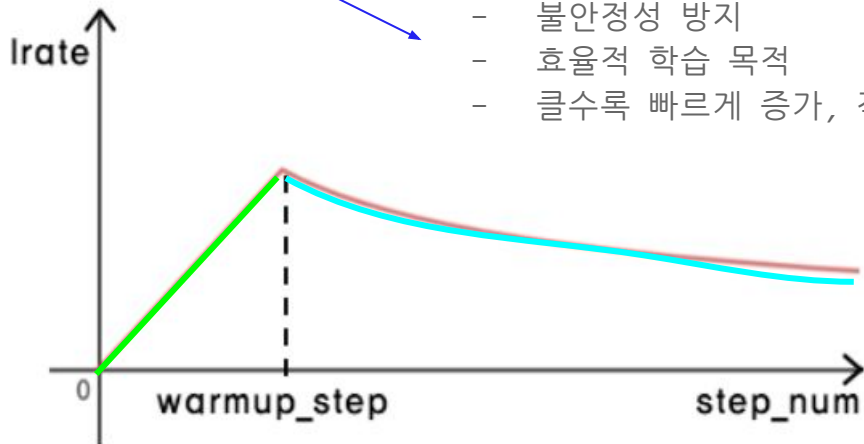
Adam

Learning rate scheduler(학습률 스케줄링): 학습률을 동적으로 변화

- 훈련 초기에는 학습률을 선형적으로 증가시키고, 이후에는 step number(단계 수)의 inverse square root(역제곱근)에 비례하여 감소
- warmup_steps = 4000

훈련 초기에 학습률을 점진적으로 증가시키는 단계

- 불안정성 방지
- 효율적 학습 목적
- 클수록 빠르게 증가, 작을수록 천천히 증가



Residual Dropout: 잔차 연결 dropout

- 각 sub layer의 출력에 dropout 적용
- 임베딩과 위치 인코딩에도 dropout 적용
- Base 모델에서 $P_{drop} = 0.1$ 사용

트랜스포머는 훈련 비용의 일부만으로
이전 모델들보다 더 높은 BLEU 점수를 달성함

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.0	$2.3 \cdot 10^{19}$	

Residual Dropout: 잔차 연결 dropout

- 각 sub layer의 출력에 dropout 적용
- 임베딩과 위치 인코딩에도 dropout 적용
- Base 모델에서 $P_{drop} = 0.1$ 사용

Dropout의 확률값을 지정하는 하이퍼파라미터

- 10% 확률로 뉴런을 drop

트랜스포머는 훈련 비용의 일부만으로
이전 모델들보다 더 높은 BLEU 점수를 달성함

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.0	$2.3 \cdot 10^{19}$	

Residual Dropout: 잔차 연결 dropout

- 각 sub layer의 출력에 dropout 적용
- 임베딩과 위치 인코딩에도 dropout 적용
- Base 모델에서 $P_{drop} = 0.1$ 사용

트랜스포머는 훈련 비용의 일부만으로
이전 모델들보다 더 높은 BLEU 점수를 달성함

기계 번역 결과와 사람이 번역한 참조 번역 간의
유사도를 측정하는 자동 평가 지표

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.0	$2.3 \cdot 10^{19}$	

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.0	$2.3 \cdot 10^{19}$	

Label Smoothing:

- $\epsilon_{ls} = 0.1$ Label Smoothing 값: 0.1 적용
- Perplexity(혼란도)를 증가시키고, accuracy (정확도)와 BLEU 점수를 개선

Label Smoothing:

- $\epsilon_{ls} = 0.1$ Label Smoothing 값: 0.1 적용
- Perplexity(혼란도)를 증가시키고, accuracy (정확도)와 BLEU 점수를 개선

<레이블을 그대로 사용하는 것이 아니라 조금 smooth하게 만들어서 정규화 시킴>

- **Hard** target(one-hot) -> **Soft** target
- $[0, 1, 0, 0] \rightarrow [0.025, 0.0925, 0.025, 0.025]$

- 실제 정확도보다 예측 확률이 큰 상태의 과잉확신을 방지
- 모델의 일반화 성능을 높여줌

06

Results

- Transformer 모델이 RNN, CNN 기반 모델보다 기계번역 작업에서 우수한 성능을 보였다.

English-to-German 번역
(WMT 2014 데이터셋)

Base 모델

- > BLEU 점수 : 25.8
- > 훈련시간 : 8개의 P100 GPU를 사용해서 3.5일
- > 기존보다 빠르고 높은 성능

Big 모델

- > BLEU 점수 : 28.4
- > 모델 크기 늘리고, Dropout 비율 0.3으로 설정

English-to-French 번역
(WMT 2014 데이터셋)

Big 모델

- > BLEU 점수 : 41.0
- > 단일 모델 기준 최고 성능
- > 훈련 비용도 기존 최고 성능 모델의 $\frac{1}{4}$ 수준

- 논문은 Transformer 모델의 하이퍼파라미터 성능을 비교한 결과를 표로 설명하고 있다.
- 설명에 들어가기에 앞서, 각 개념을 짚고 넘어가자.

 N

Number of Layers : 트랜스포머의 Encoder/Decoder 블록 수를 의미한다.

ex) $N=5$ 라면, Encoder와 Decoder가 각각 5개의 레이어를 가진다는 뜻.

역할 : 레이어가 많아질수록 모델의 깊이가 증가하고, 더 복잡한 패턴을 학습할 수 있다.

 d_{model}

Model Dimension : 입력 단어를 벡터로 변환할 때 사용되는 임베딩 벡터의 차원.

ex) $d_{model}=512$ 라면, 각 단어가 512차원 벡터로 표현된다.

역할 : 모델이 표현할 수 있는 정보의 크기를 결정한다. 커지면 성능이 향상되지만 비용도 증가.

 h

Number of Attention Heads: Multi-Head Attention에서 쓰이는 헤드의 개수.

역할 : 여러 헤드가 데이터를 다양한 방식으로 분석해서 더 풍부한 표현을 학습.

 d_k, d_v

Key/Value Dimensions: Multi-Head Attention에서 Key와 Value 벡터의 차원 크기.

Key/Value 차원이 작아지면 계산 효율성이 증가하지만, 너무 작으면 성능이 저하된다.

 P_{drop}

Dropout Probability : 학습 중 뉴런 출력을 랜덤하게 제거할 확률. 0.1을 값으로 주면 뉴런의 10%를 학습 중에 제거한다. 오버피팅 방지를 위한 드롭아웃 비율이라고 보면 된다.

역할 : 오버피팅 방지 및 일반화 성능 향상

ϵ_{ls}

Label Smoothing: 레이블 스무딩은 정답 레이블의 확률을 살짝 낮추고, 나머지 클래스에 확률을 조금 나눠주는 기법이다. 예를 들어, 0.1을 값으로 주면 정답 레이블 확률을 0.9로, 나머지 클래스에 0.1을 분산시키는 식이다.

역할 : 모델이 과도하게 정답 레이블에 의존하지 않도록 돕고, 일반화 성능을 향상시킨다.

train
steps

모델이 학습을 위해 데이터를 몇 번 업데이트 했는지를 나타낸다. 훈련 스텝이 많을수록 더 많은 데이터로 학습하지만, 과적합 위험이 증가할 수 있다.

PPL
(dev)

Perplexity : 언어 모델의 성능을 측정하는 척도. 낮을수록 모델이 더 정확한 예측을 한다는 뜻이다. 모델의 품질을 정량적으로 평가하는 역할을 한다.

BLEU
(dev)

BLEU Score : 번역 모델의 성능을 평가하는 점수. 높을수록 번역 품질이 좋음을 나타낸다. 번역 작업에서 모델 성능을 직접적으로 평가하는 역할을 한다.

params
 $\times 10^6$

파라미터 수. 모델에서 학습되는 가중치(파라미터)의 총 개수를 백만 단위로 표현했다. 모델 크기를 평가한다.

d_{ff}

Feed-Forward Dimension : FFN 내부에서 데이터 차원을 확장하는 중간 레이어 차원.

피드포워드 네트워크(FFN)란? 앞에도 나왔지만 여기서 간단히 또 짚어보자면

Fully Connected Layer(완전연결층)

트랜스포머의 인코더, 디코더 블록에는 Multi-Head Attention 이후에 Feed-Forward Network가 들어간다. **입력벡터를 비선형적으로 변환하기 위해서다.**

더 자세히 설명하자면, **Multi-Head Attention에서 계산된 결과는 그대로 사용할 수 없다.** Attention 결과를 더 잘 변환하고, 학습 가능한 패턴을 추출하기 위해 FFN을 사용한다.

Multi-Head Attention은 각 입력벡터(단어 벡터)에 대해 문맥 정보를 계산한다. 즉, 입력 단어가 문장 내에서 다른 단어들과 어떤 관계에 있는지 학습한다.

단어 간의 관계를 계산하는 데 특화됐지만, 개별 단어 벡터를 풍부하게 변환하는 데는 **한계가** 있다.

예를 들어 rabbit과 sleep의 관계는 잘 표현하지만, rabbit 자체의 특성은 학습에 한계가 있다.

그래서 Multi-Head Attention 이후에 FFN을 추가로 적용하면

Attention이 계산한 문맥 정보를 바탕으로 단어 벡터를 개별적으로 비선형적으로 변환해 더 풍부하게 학습하고, 문맥 속 단어의 의미를 더 깊이 이해할 수 있게 된다.

참고 : 선형 변환은 단순히
입출력 간의 선형 관계만 표현할
수 있다.
비선형 변환은 더 복잡한 관계를
모델링하는 과정이다.

공식

$$FFN(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

X : Multi-Head Attention의 출력
W1, W2 : 학습 가능한 가중치 행렬
b1, b2 : 학습가능한 편향(bias)
ReLU : 비선형 활성화 함수

d_{ff} 를 그래서 왜 쓰는 걸까?

입력데이터(벡터)를 변화하는 것만으로는 충분히 복잡한 패턴을 학습할 수 없다.

FFN은 차원을 d_{ff} 로 확장해서 더 많은 계산을 수행하고, 데이터를 비선형적으로 변환한다.
결과적으로, 모델이 더 복잡한 관계와 패턴을 학습할 수 있게 된다.

FFN의 용량을 결정하는 요소라고 볼 수 있다.

예를 들어 모델 디멘전이 512, d_{ff} 가 2048 라면, 입력 벡터의 크기가 512차원이 된다.
FFN의 첫번째 레이어가 512 차원 데이터를 2048차원으로 확장시킨다.

이후 ReLU 함수로 비선형 변환을 적용하고, 두 번째 레이어에서 다시 512차원으로 줄인다.
이 과정을 거쳐 입력 데이터에 대한 더 복잡한 표현을 모델이 학습할 수 있게 된다.

- Base 모델을 기준으로 다양한 변형 모델((A), (B), (C) 등)을 실험한 결과를 비교하고 있다.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)	<ul style="list-style-type: none"> Base 모델은 기본 설정으로 빠르고 효율적이고, 번역 성능이 준수한 모델이다. Big 모델은 크기를 키워서 더 높은 성능을 얻었지만, 계산 비용이 더 높은 모델이다. <p>Base 모델과 Big 모델만 비교해서 먼저 보면</p> <p>Big 모델이 약 2배 정도씩 더 큰 하이퍼파라미터가 적용됐지만, 결과는 비슷한 걸 볼 수 있다.</p>											
(B)												
(C)												
(D)												
(E)												
big	6	1024	4096	16			0.3		300K	4.33	26.4	213

6.2 Model Variations

기본적으로 사용한
하이퍼파라미터

Head 디멘전을
바꾸면서 key와
value 디멘전도
바뀜.

헤드 상관 없이 key
디멘전 줄이기. 파라미터
수가 줄어서 BLEU도
줄었다.

모델 차원 크기를
늘리거나, FFN 내부
레이어 차원 높였을 때
성능이 좋아진다.

드롭 아웃, 레이블
스무딩도 효과가 있다.

위치 정보를 주는
positional embedding이
기존의 sin, cos 함수를
이용한 인코딩에 비해
성능에 악영향을 미치는지
봤을 때, 해당 요소가
미치는 영향은 미미하다는
걸 알 수 있다.

Head가 8, 16일 때
성능이 가장 좋음

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
							0.0			5.77	24.6	
(D)							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	4.33	26.4	213

07

conclusion

- Transformer는 전적으로 Attention 매커니즘 기반으로 설계된 최초의 모델
- 기존의 RNN, CNN 기반 번역 모델에 비교했을 때 속도, 성능 모두 더 우수하다.
- 특히 번역 성능에서는 기존 기록을 갱신할 정도.
- 앞으로 이미지, 오디오, 비디오 등 다른 분야에도 적용 가능성이 크다.
- 더 큰 입력데이터를 다루기 위해 local Attention 같은 방법을 연구할 예정.