KOREA UNIVERSITY

DSBA Transformer survey paper study

A Survey of Transformers

#6: Architecture-Level Variants

arXiv preprint

DSBA
Data Science & Business Analytics

고려대학교 산업경영공학과

Data Science & Business Analytics Lab

이유경, 김명섭, 윤훈상, 김지나, 허재혁, 김수빈

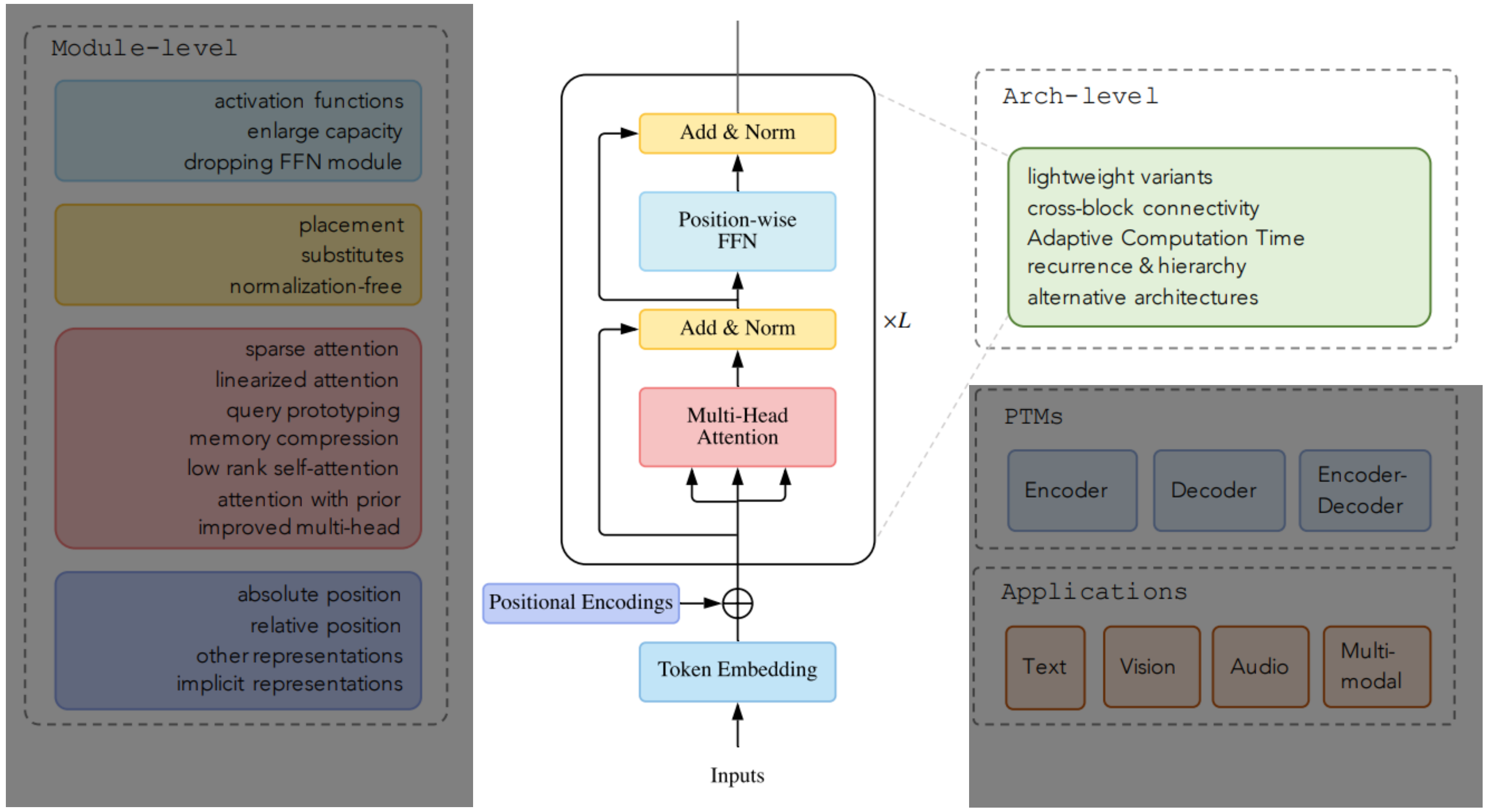
발표자 : 김명섭

06

Taxonomy of Transformers

Architecture-Level Variant

- 이번 발표는 Architecture-Level Variant에 대해 진행됨
- Architecture-Level Variant는 Module 단위를 넘어 Transformer Encoder / Decoder 구조에 대한 전반적인 변화를 줌



<Architecture-Level Transformer Variant>

06 Taxonomy of Transformers

Architecture-Level Variant

Architecture-Level Variant는 크게 아래와 같은 목적으로 구분

- Adapting Transformer to Be Lightweight
  - Transformer를 Module 단위 이상의 수정을 통해 경량화, Mobile Setting 등에 활용될 수 있도록 함
- Strengthening Cross-Block Connectivity
  - Transformer Encoder와 Decoder간의 정보 전달을 강화
- Adaptive Computation Time
  - Input을 처리하기 위해 고정된 시간이 필요했던 Transformer의 연산을 개선, Input에 따라 가변적인 연산을 수행
- Transformers with Divide-and-Conquer Strategies
  - Long-range Context를 처리하기 위해 Input Sequence를 Decompose하여 처리

06

Taxonomy of Transformers

Architecture-Level Variant

- Architecture-Level Variant
  - Adapting Transformer to Be Lightweight
    - Lite Transformer (ICLR, 2019, Citation: 51, Massachusetts Institute of Technology / Shanghai Jiao Tong University)
    - Funnel Transformer (NeurIPS, 2020, Citation: 30, Carnegie Mellon University / Google AI Brain Team)
    - DeLight (ICLR, 2021, Citation: 2, University of Washington / Facebook AI / Allen Institute for AI)
  - Strengthening Cross-Block Connectivity
    - RealFormer (arXiv, 2020, Citation: 5, Google Research)
    - Predictive Attention Transformer (ICLR (rejected), 2021, Citation: 1)
    - Transparent Attention (EMNLP, 2018, Citation: 55, Google AI)
    - Feedback Transformer (ICLR (rejected), 2021, Citation: 4)
  - Adaptive Computation Time
    - Universal Transformer (ICLR, 2019, Citation: 331, University of Amsterdam / DeepMind / Google Brain)
    - Conditional Computation Transformer (arXiv, 2020, Citation: 7, Google Research)
    - DeeBERT (ACL, 2020, Citation: 49, David R. Cheriton School of Computer Science / University of Waterloo, Vector Institute for Artificial Intelligence)
    - PABEE (NeurIPS, 2020, Citation: 29, Beihang University / University of California San Diego, Microsoft Research Asia)
    - SENTEE / TOKEE (ACL, 2021, Citation: 1, Fudan University)
    - Sun et al. (arXiv (under review), 2021, Citation: 1, Fudan University / Huawei Poisson Lab)
  - Transformers with Divide-and-Conquer Strategies
    - Recurrent
      - Transformer-XL (ACL, 2019, Citation: 1190, Carnegie Mellon University / Google Brain)
      - Compressive Transformer (ICLR, 2020, Citation: 91, DeepMind / University College London)
      - Memformer (ICLR (rejected), 2021, Citation: 0)
      - Yoshida et al. (arXiv, 2020, Citation: 0, Toyota Technological Institute / University of Chicago)
      - ERNIE-Doc (ACL, 2021, Citation: 1, Baidu Inc.)
    - Hierarchy
      - Miculicich et al. (EMNLP, 2018, Citation: 135, Idiap Research Institute / Ecole Polytechnique Federale de Lausanne)
      - HIBERT (ACL, 2019, Citation: 143, Microsoft Research Asia)
      - Liu and Lapata (ACL, 2019, Citation: 124, University of Edinburgh)
      - Hi-Transformer (ACL, 2021, Citation: 1, Tsinghua University / Microsoft Research Asia)
      - TENER (arXiv, 2019, Citation: 39, Fudan University)
      - TNT (arXiv (under review), 2021, Citation: 55, Huawei Technologies / ISCAS & UCAS)

06

Taxonomy of Transformers

Architecture-Level Variant

- Architecture-Level Variant
  - Adapting Transformer to Be Lightweight
    - Lite Transformer (ICLR, 2019, Citation: 51, Massachusetts Institute of Technology / Shanghai Jiao Tong University)
    - Funnel Transformer (NeurIPS, 2020, Citation: 30, Carnegie Mellon University / Google AI Brain Team)
    - DeLight (ICLR, 2021, Citation: 2, University of Washington / Facebook AI / Allen Institute for AI)
  - Strengthening Cross-Block Connectivity
    - RealFormer (arXiv, 2020, Citation: 5, Google Research)
    - Predictive Attention Transformer (ICLR (rejected), 2021, Citation: 1)
    - Transparent Attention (EMNLP, 2018, Citation: 55, Google AI)
    - Feedback Transformer (ICLR (rejected), 2021, Citation: 4)
  - Adaptive [illegible]mputation Time
    - U[illegible]vers[illegible] ([illegible]2019, [illegible]itation: 331, U[illegible]vers[illegible] o[illegible] Amsterdam / De[illegible] / [illegible]
    - C[illegible]diti[illegible]al Co[illegible]put[illegible]ion T[illegible]rm[illegible] (arXiv, 202[illegible] Citat[illegible]: 7, [illegible]oogle Res[illegible]arch)
    - D[illegible]BE[illegible] (ACL, 202[illegible] C[illegible]ion[illegible]9, [illegible]avid R. C[illegible]iton Schoo[illegible] Computer [illegible]cience / [illegible]ive[illegible]ity [illegible] Wat[illegible]o[illegible] Vec[illegible] In[illegible]tute for A[illegible]icial Intelligence)
    - P[illegible]BEE ([illegible]S, 20[illegible]29, Beihang University / Univ[illegible]rsity of Ca[illegible]ornia San [illegible] Mi[illegible] Rese[illegible] Asia)
    - SENTEE / TOKEE (ACL, 2021, Citation: 1, Fudan University)
    - Sun et al. (arXiv (under review), 2021, Citation: 1, Fudan University / Huawei Poisson Lab)
  - Transformers wit[illegible]
    - Recurrent
      - Transformer-XL (ACL, 2019, Citation: 1190, Carnegie Mellon University / Google Brain)
      - Compressive Transformer (ICLR, 2020, Citation: 91, DeepMind / University College London)
      - Memformer (ICLR (rejected), 2021, Citation: 0)
      - Yoshida et al. (arXiv, 2020, Citation: 0, Toyota Technological Institute / University of Chicago)
      - ERNIE-Doc (ACL, 2021, Citation: 1, Baidu Inc.)
    - Hierarchy
      - Miculicich et al. (EMNLP, 2018, Citation: 135, Idiap Research Institute / Ecole Polytechnique Federale de Lausanne)
      - HIBERT (ACL, 2019, Citation: 143, Microsoft Research Asia)
      - Liu and Lapata (ACL, 2019, Citation: 124, University of Edinburgh)
      - Hi-Transformer (ACL, 2021, Citation: 1, Tsinghua University / Microsoft Research Asia)
      - TENER (arXiv, 2019, Citation: 39, Fudan University)
      - TNT (arXiv (under review), 2021, Citation: 55, Huawei Technologies / ISCAS & UCAS)

Total 24 Papers…

(Except for Alternative Architecture)

06 Taxonomy of Transformers

Architecture-Level Variant

- Architecture-Level Variant
  - Adapting Transformer to Be Lightweight
    - Lite Transformer (ICLR, 2019, Citation: 51, Massachusetts Institute of Technology / Shanghai Jiao Tong University)
    - Funnel Transformer (NeurIPS, 2020, Citation: 30, Carnegie Mellon University / Google AI Brain Team)
    - DeLight (ICLR, 2021, Citation: 2, University of Washington / Facebook AI / Allen Institute for AI)
  - Strengthening Cross-Block Connectivity
    - RealFormer (arXiv, 2020, Citation: 5, Google Research)
    - Predictive Attention Transformer (ICLR (rejected), 2021, Citation: 1)
    - Transparent Attention (EMNLP, 2018, Citation: 55, Google AI)
    - Feedback Transformer (ICLR (rejected), 2021, Citation: 4)
  - Adaptive Computation Time
    - Universal Transformer (ICLR, 2019, Citation: 331, University of Amsterdam / DeepMind / Google Brain)
    - Conditional Computation Transformer (arXiv, 2020, Citation: 7, Google Research)
    - DeeBERT (ACL, 2020, Citation: 49, David R. Cheriton School of Computer Science / University of Waterloo, Vector Institute for Artificial Intelligence)
    - PABEE (NeurIPS, 2020, Citation: 29, Beihang University / University of California San Diego, Microsoft Research Asia)
    - SENTEE / TOKEE (ACL, 2021, Citation: 1, Fudan University)
    - Sun et al. (arXiv (under review), 2021, Citation: 1, Fudan University / Huawei Poisson Lab)
  - Transformers with Divide-and-Conquer Strategies
    - Recurrent
      - Transformer-XL (ACL, 2019, Citation: 1190, Carnegie Mellon University / Google Brain)
      - Compressive Transformer (ICLR, 2020, Citation: 91, DeepMind / University College London)
      - Memformer (ICLR (rejected), 2021, Citation: 0)
      - Yoshida et al. (arXiv, 2020, Citation: 0, Toyota Technological Institute / University of Chicago)
      - ERNIE-Doc (ACL, 2021, Citation: 1, Baidu Inc.)
    - Hierarchy
      - Miculicich et al. (EMNLP, 2018, Citation: 135, Idiap Research Institute / Ecole Polytechnique Federale de Lausanne)
      - HIBERT (ACL, 2019, Citation: 143, Microsoft Research Asia)
      - Liu and Lapata (ACL, 2019, Citation: 124, University of Edinburgh)
      - Hi-Transformer (ACL, 2021, Citation: 1, Tsinghua University / Microsoft Research Asia)
      - TENER (arXiv, 2019, Citation: 39, Fudan University)
      - TNT (arXiv (under review), 2021, Citation: 55, Huawei Technologies / ISCAS & UCAS)

06 Adapting Transformer to Be Lightweight

Lite Transformer (ICLR, 2019, Citation: 51)

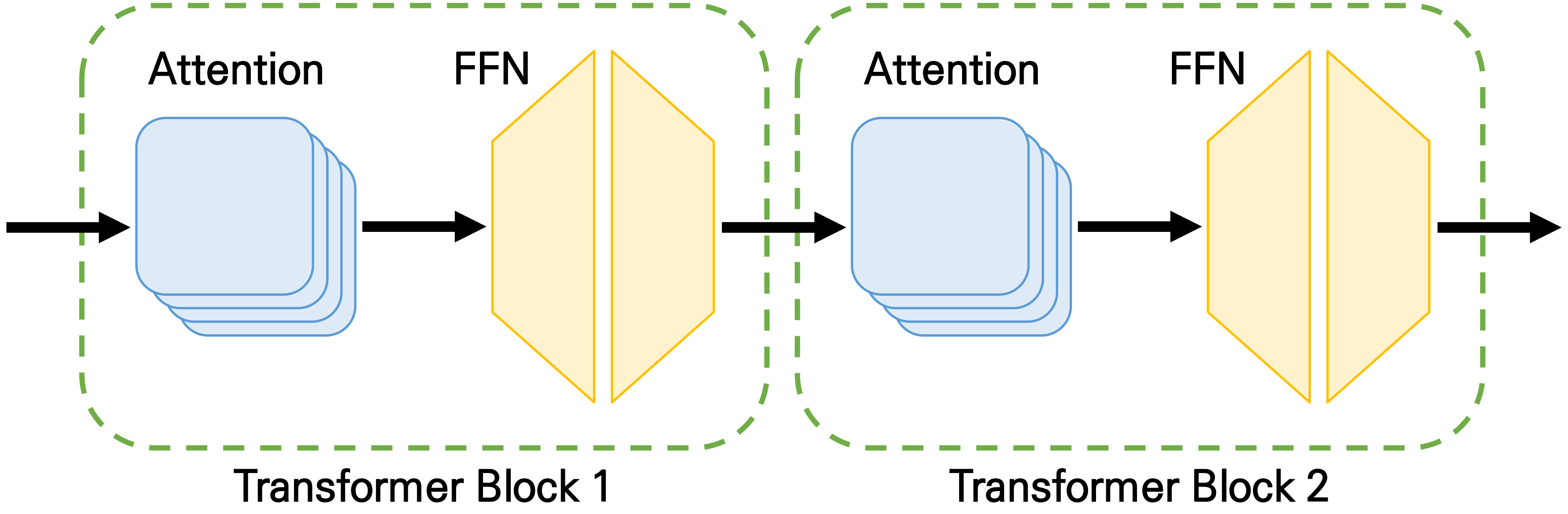
- Lite Transformer
  - Hardware Resource와 Battery의 제약이 존재하는 Mobile 환경에서 Transformer를 동작할 수 있도록 설계
  - Lite Transformer의 Main Idea는 $Long - Short\ Range\ Attention$ (LSRA)
    - 일부의 Head를 이용해 Convolution연산을 수행하여 Local Context를 Modeling
    - 다른 일부의 Head를 이용해 Attention을 통해 Long-Distance Relationship을 Modeling
    - Self-Attention을 수행하는 Head의 수가 감소하는 만큼 Computational Complexity가 감소
  - FFN의 경우 어떠한 Context Modeling을 수행하지 않음에도 매우 많은 Computational Complexity를 요구 $(8nd^2)$
    - Flattening을 수행하여 FFN의 Complexity를 감소

06

Adapting Transformer to Be Lightweight

Lite Transformer (ICLR, 2019, Citation: 51)

- Lite Transformer
  - Transformer의 Block은 Multi-Head Self-Attention과 Position-Wise FFN으로 구성

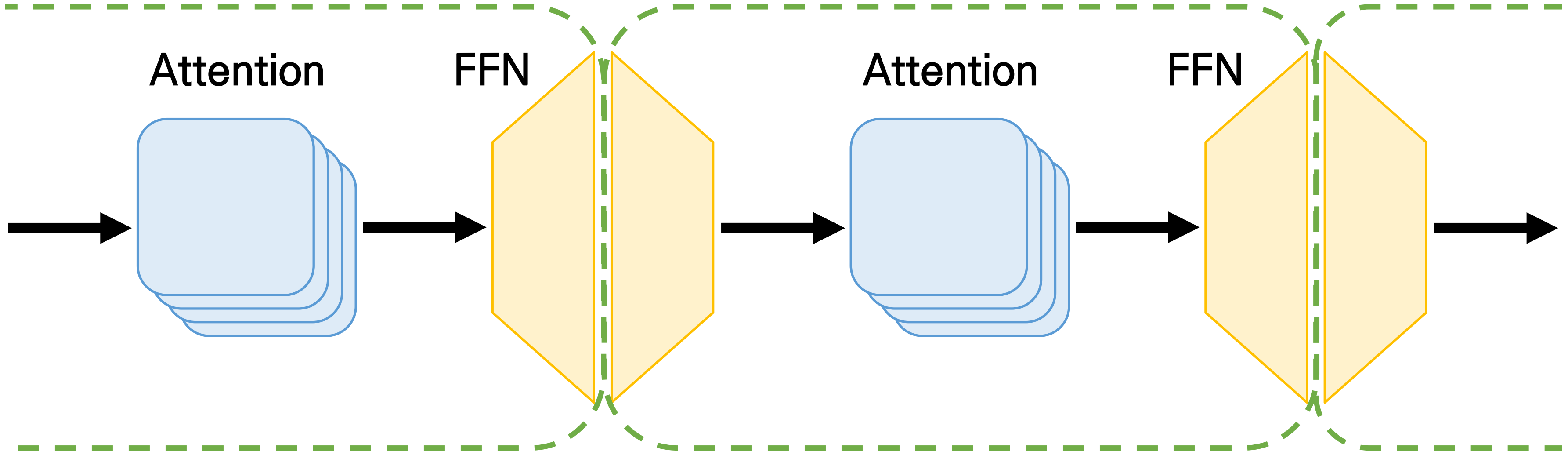


06

Adapting Transformer to Be Lightweight

Lite Transformer (ICLR, 2019, Citation: 51)

- Lite Transformer
  - Transformer의 Block은 아래와 같은 표현으로도 볼 수 있음



Different View of Transformer Block

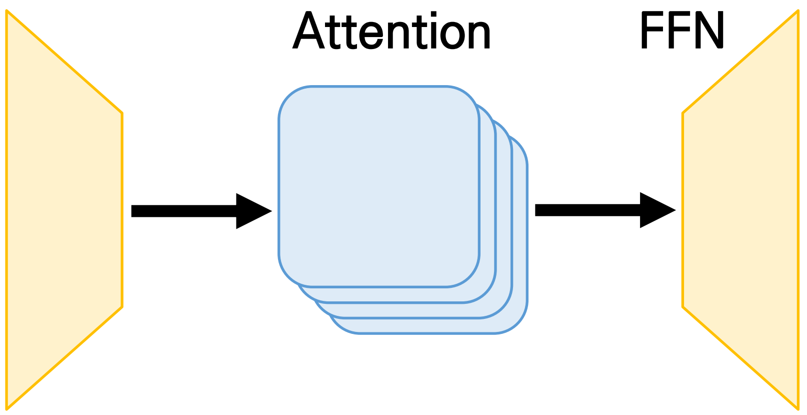
06

Adapting Transformer to Be Lightweight

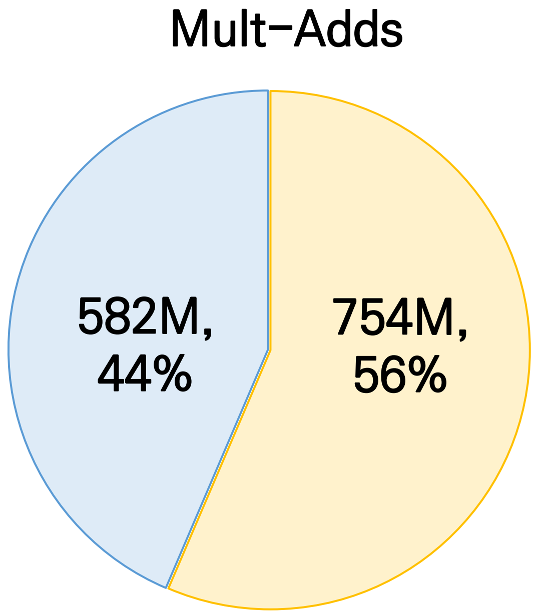
Lite Transformer (ICLR, 2019, Citation: 51)

[Presentation] Lite Transformer, ICLR, 2019

- Lite Transformer
  - Multi-Head Self-Attention과 Position-Wise FFN의 계산 복잡도는 아래와 같음
    - Multi-Head Self-Attention : $\boldsymbol{O}(4nd^2 + 2n^2d)$
    - Position-Wise FFN : $\boldsymbol{O}(8n^2d)$
  - Vanilla Transformer에서 연산의 비율은 오른쪽 그림과 같음



Base Transformer Block



Computation Proportion

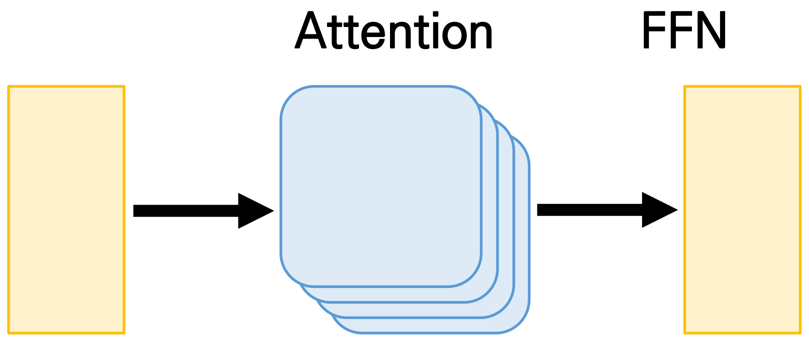
06

Adapting Transformer to Be Lightweight

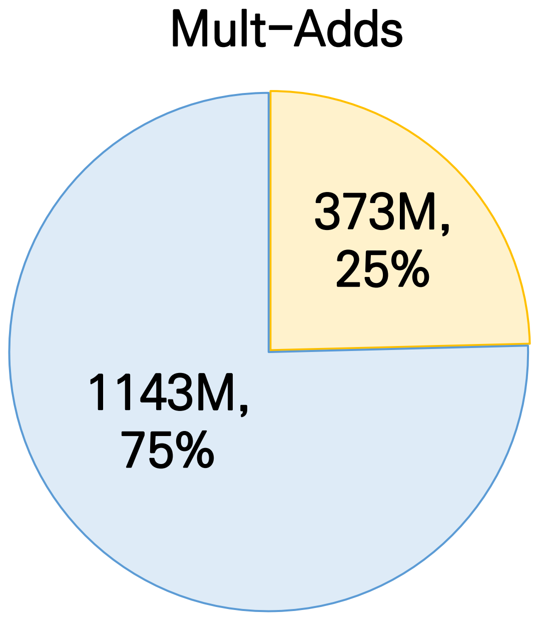
Lite Transformer (ICLR, 2019, Citation: 51)

[Presentation] Lite Transformer, ICLR, 2019

- Lite Transformer
    - Position-Wise FFN의 경우 어떠한 Context Modeling을 수행하지 않음에도, 많은 Computation을 차지하고 있음
    - FFN을 Flatten하게 될 경우, 많은 양의 연산을 감소시킬 수 있음



Base Transformer Block

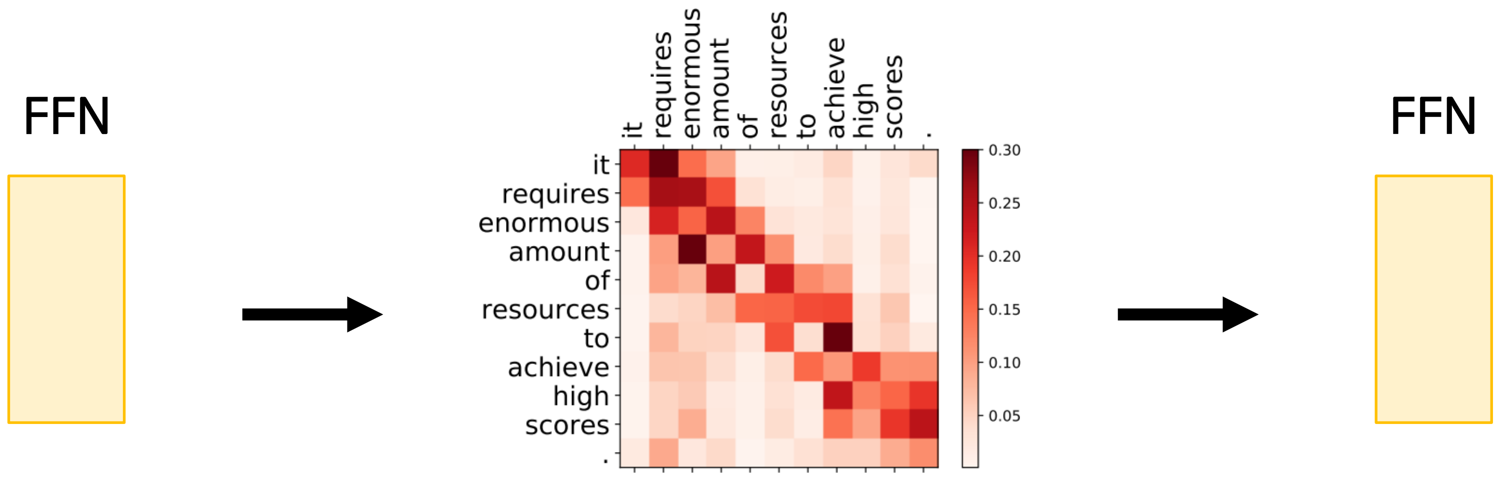


Computation Proportion

Adapting Transformer to Be Lightweight

Lite Transformer (ICLR, 2019, Citation: 51)

- Lite Transformer
  - Multi-Head Self-Attention에 존재하는 Attention Weight를 시각화 하면 아래와 같은 결과를 볼 수 있음
  - 기존의 Multi-Head Self-Attention의 경우 Diagonal Local Context와 Sparse한 Global Context를 함께 Modeling



Visualization of Attention Weight

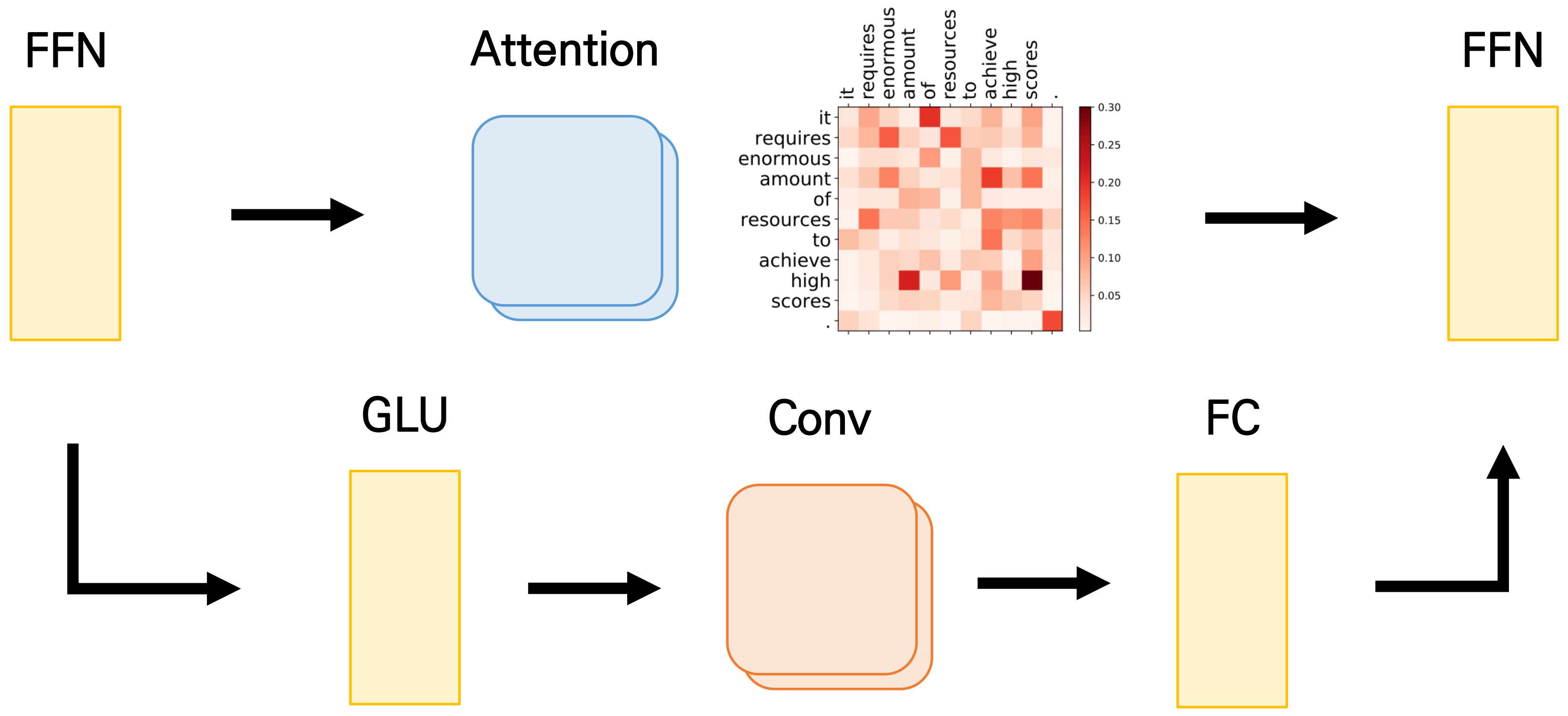
06

Adapting Transformer to Be Lightweight

Lite Transformer (ICLR, 2019, Citation: 51)

[Presentation] Lite Transformer, ICLR, 2019

- Lite Transformer
  - Multi-Head Self-Attention의 Head 수를 감소시키고 Diagonal Local Context를 반영하는 Conv를 추가하여 연산량을 감소
  - Multi-Head Self-Attention은 Sparse Global Context를 반영



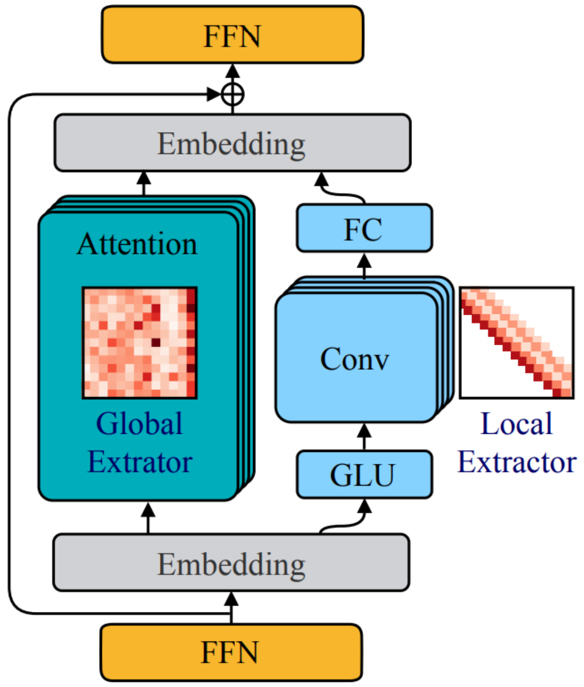
Long-Short Range Attention (LSRA)

Adapting Transformer to Be Lightweight

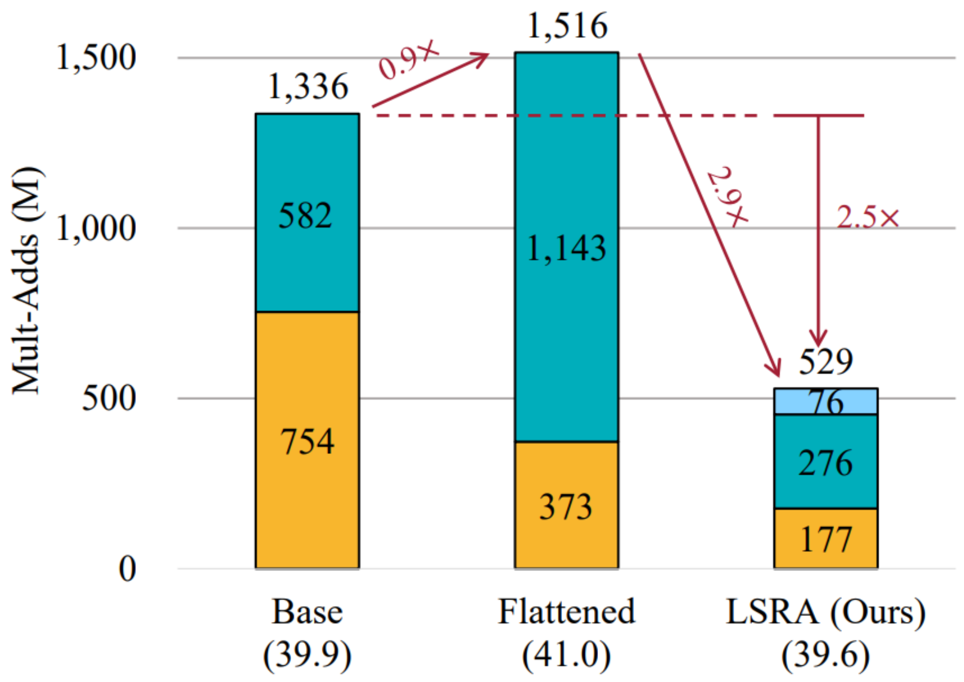
Lite Transformer (ICLR, 2019, Citation: 51)

[Presentation] Lite Transformer, ICLR, 2019

- Lite Transformer



Lite Transformer

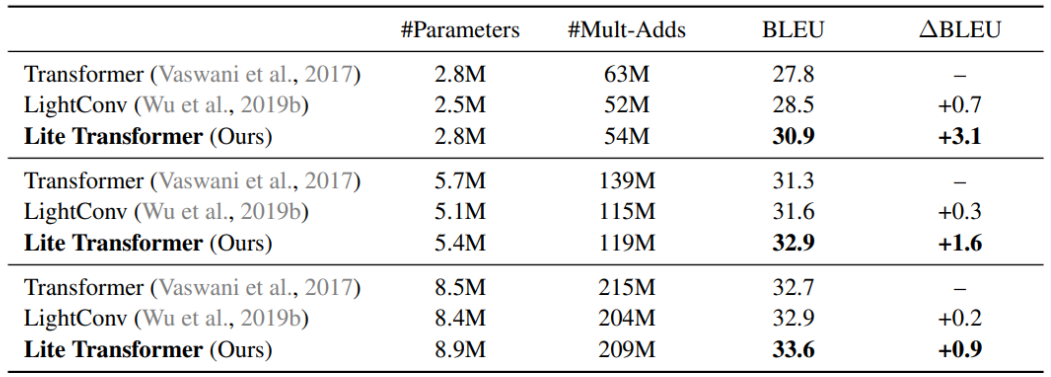


Computational Complexity & BLEU Score

06 Adapting Transformer to Be Lightweight

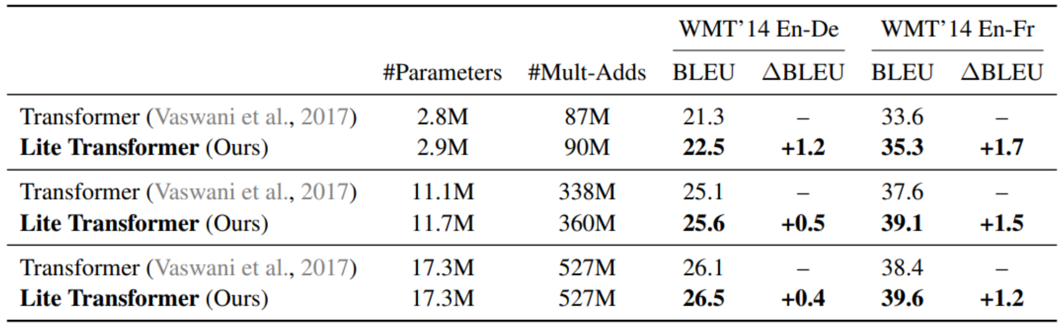
Lite Transformer (ICLR, 2019, Citation: 51)

- Lite Transformer
  - IWSLT'14, WMT'14 Dataset에서 Mobile Setting과 Similar Complexity에서 Baseline 대비 뛰어난 성능을 확인



	#Parameters	#Mult-Adds	BLEU	ΔBLEU
Transformer (Vaswani et al., 2017)	2.8M	63M	27.8	–
LightConv (Wu et al., 2019b)	2.5M	52M	28.5	+0.7
Lite Transformer (Ours)	2.8M	54M	**30.9**	**+3.1**
Transformer (Vaswani et al., 2017)	5.7M	139M	31.3	–
LightConv (Wu et al., 2019b)	5.1M	115M	31.6	+0.3
Lite Transformer (Ours)	5.4M	119M	**32.9**	**+1.6**
Transformer (Vaswani et al., 2017)	8.5M	215M	32.7	–
LightConv (Wu et al., 2019b)	8.4M	204M	32.9	+0.2
Lite Transformer (Ours)	8.9M	209M	**33.6**	**+0.9**

Table 1: Results on IWSLT'14 De-En. Our Lite Transformer outperforms the transformer (Vaswani et al., 2017) and the Lightweight convolution network (Wu et al., 2019b) especially in mobile settings.



			WMT'14 En-De		WMT'14 En-Fr	
	#Parameters	#Mult-Adds	BLEU	ΔBLEU	BLEU	ΔBLEU
Transformer (Vaswani et al., 2017)	2.8M	87M	21.3	–	33.6	–
Lite Transformer (Ours)	2.9M	90M	**22.5**	**+1.2**	**35.3**	**+1.7**
Transformer (Vaswani et al., 2017)	11.1M	338M	25.1	–	37.6	–
Lite Transformer (Ours)	11.7M	360M	**25.6**	**+0.5**	**39.1**	**+1.5**
Transformer (Vaswani et al., 2017)	17.3M	527M	26.1	–	38.4	–
Lite Transformer (Ours)	17.3M	527M	**26.5**	**+0.4**	**39.6**	**+1.2**

Table 2: Results on WMT'14 En-De and WMT'14 En-Fr. Our Lite Transformer improves the BLEU score over the transformer under similar Mult-Adds constraints.

06

Strengthening Cross-Block Connectivity

Transparent Attention (EMNLP, 2018, Citation: 55)

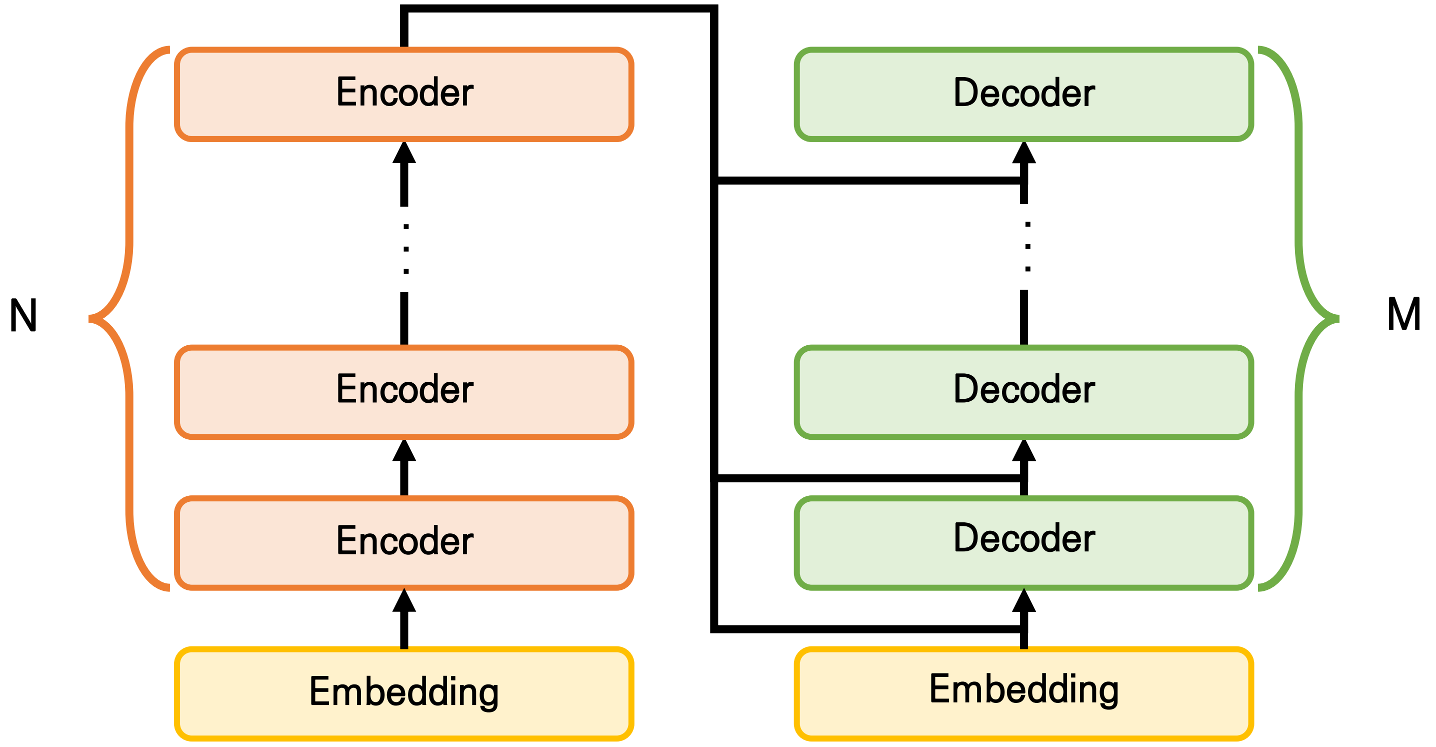
- Transparent Attention
  - Seq2Seq과 Transformer 등의 Model이 Neural Machine Translation (NMT)에서 State-of-the-art의 성능을 보임 (2018년 기준)
  - 이들 Model은 많은 Parameter를 보유하고 있으나, Vision Domain의 Model보다 Shallow한 구조를 가지고 있음
  - NMT Model이 Deep한 구조를 갖기 힘든 이유는 아래와 같음
    - Decoder의 마지막 Layer에서 Encoder의 첫 번째 Layer까지 Gradient를 전파하기 위해 많은 Step과 Parameter를 거쳐야 함
    - Encoder와 Decoder를 연결하는 부분의 Bottleneck을 거쳐야 함
  - 해당 논문에서는 Encoder의 각 Layer와 Decoder의 각 Layer간 Attention을 연결하는 Transparent Attention을 제안
  - Vanilla Transformer 대비 2~3배 깊은 Transformer Model을 구현

06

Strengthening Cross-Block Connectivity

Transparent Attention (EMNLP, 2018, Citation: 55)

- Transparent Attention
    - Vanilla Transformer의 경우 Encoder와 Decoder간 그림과 같은 연결 구조를 가지고 있음
    - 모든 Decoder Layer는 마지막 Encoder Layer의 Representation을 이용해 Encoder-Decoder Attention을 수행



Encoder-Decoder Connection

Encoder-Decoder Attention

$$\{\boldsymbol{h_t^N} \mid \boldsymbol{t = 1, \ldots, T}\}$$

$T: Time\ Step, Token$

$N: Encoder\ Layers$

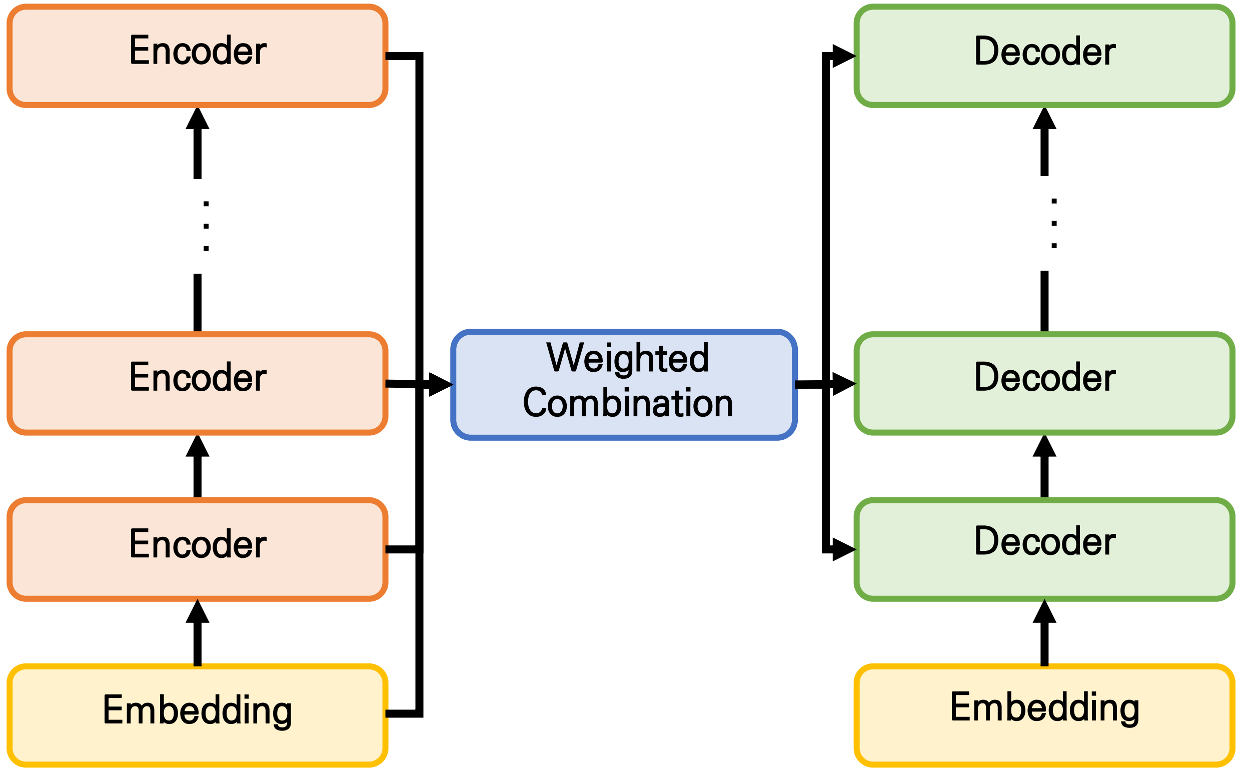
$M: Decoder\ Layers$

$h_t^i: Hidden\ Representation\ of\ i-th\ Encoder$

06 Strengthening Cross-Block Connectivity

Transparent Attention (EMNLP, 2018, Citation: 55)

- Transparent Attention
  - Transparent Attention에서는 다음과 같은 연결 구조를 도입
  - 각 Encoder Layer의 Representation에 Softmax를 이용한 Weighted Combination을 적용하여 Decoder에 전달



Transparent Attention

Weighted Combination of Encoder

$$s_{i,j} = \frac{e^{W_{i,j}}}{\sum_{k=0}^{N} e^{W_{k,j}}}, j = 1, \ldots, M$$

$$z_t^j = \sum_{i=1}^{N+1} s_{i,j} h_t^i, \quad t = 1, \ldots, T \quad j = 1, \ldots, M$$

$W: \big((N+1) \times M\big)$ *Weight Matrix, Learnable*

T: *Time Step, Token*

N: *Encoder Layers*

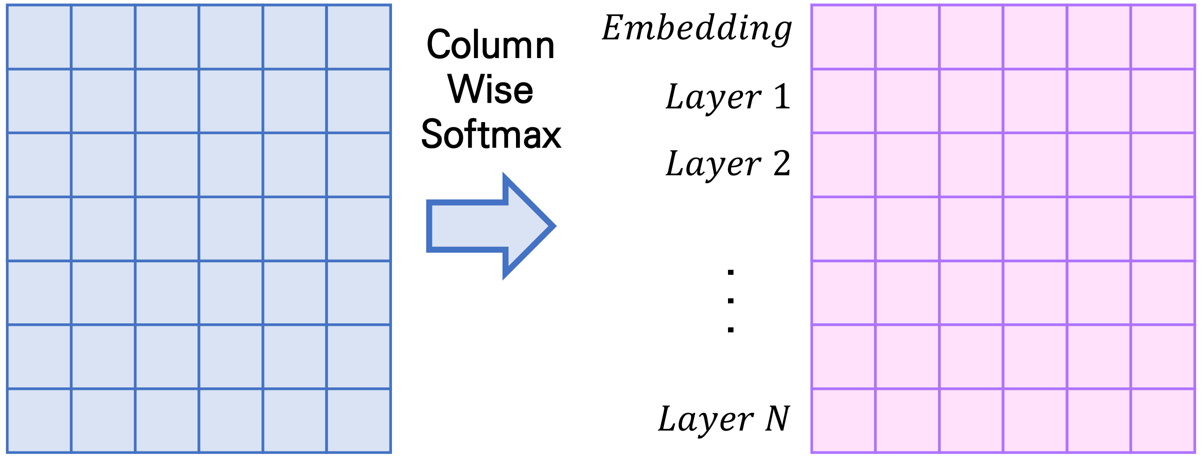
M: *Decoder Layers*

06

Strengthening Cross-Block Connectivity

Transparent Attention (EMNLP, 2018, Citation: 55)

- Transparent Attention
  - Weight Matrix는 ((N+1), M) 크기의 행렬로, Column Wise Softmax를 적용하여 각 Encoder Representation에 대한 가중치를 계산
  - 계산된 가중치는 각 Encoder의 Representation에 곱해진 후, 모든 Encoder Representation을 더하여 Decoder에서 활용
  - Weight Matrix는 Learnable Matrix로 Training 중 Update됨



Weighted Combination Matrix $(\boldsymbol{W})$
$((N+1) \times M)$

Softmax Matrix $(\boldsymbol{S})$
$((N+1) \times M)$

Weighted Combination of Encoder

$$s_{i,j} = \frac{e^{W_{i,j}}}{\sum_{k=0}^{N} e^{W_{k,j}}}, j = 1, \dots, M$$

$$z_t^j = \sum_{i=1}^{N+1} s_{i,j} h_t^i, \quad t = 1, \dots, T \quad j = 1, \dots, M$$

$W: ((N+1) \times M)$ *Weight Matrix, Learnable*

$T:$ *Time Step, Token*

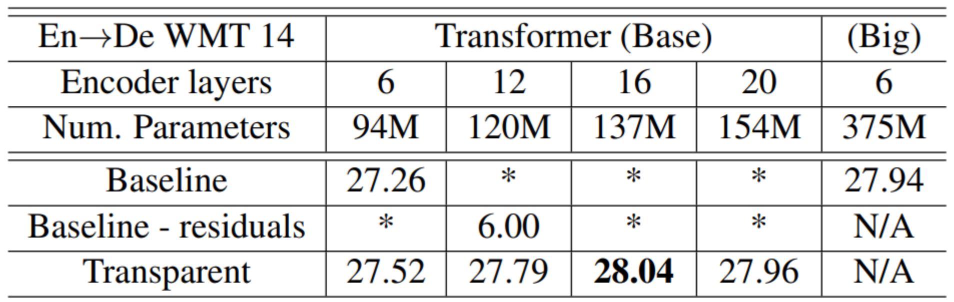
$N:$ *Encoder Layers*

$M:$ *Decoder Layers*

06 Strengthening Cross-Block Connectivity

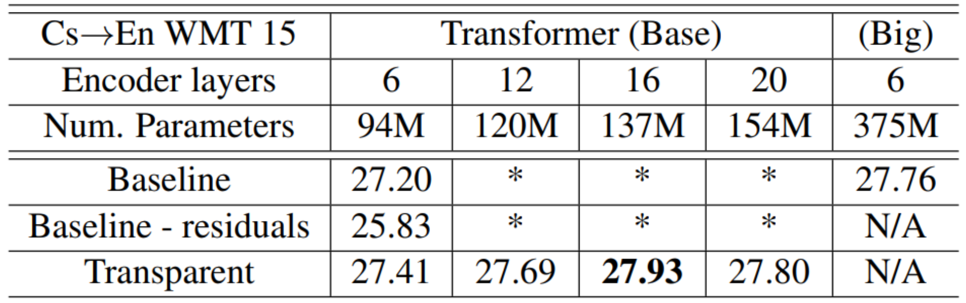
Transparent Attention (EMNLP, 2018, Citation: 55)

- **Transparent Attention**
  - NMT Task (WMT 14, WMT 15)에 대해 보다 깊은 Transformer를 안정적으로 학습할 수 있다는 것을 보임
  - Transparent Attention을 적용한 깊은 구조의 Transformer가 Baseline Model보다 높은 성능을 기록함



En→De WMT 14	Transformer (Base)				(Big)
Encoder layers	6	12	16	20	6
Num. Parameters	94M	120M	137M	154M	375M
Baseline	27.26	*	*	*	27.94
Baseline - residuals	*	6.00	*	*	N/A
Transparent	27.52	27.79	**28.04**	27.96	N/A

Table 1: BLEU scores on En→De newstest 2014 with Transformers. * indicates that a model failed to train.



Cs→En WMT 15	Transformer (Base)				(Big)
Encoder layers	6	12	16	20	6
Num. Parameters	94M	120M	137M	154M	375M
Baseline	27.20	*	*	*	27.76
Baseline - residuals	25.83	*	*	*	N/A
Transparent	27.41	27.69	**27.93**	27.80	N/A

Table 2: BLEU scores Cs→En newstest 2015 with Transformers. * indicates that a model failed to train.

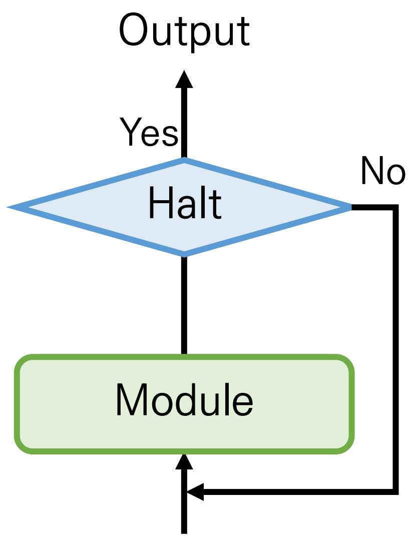
06

Adaptive Computation Time

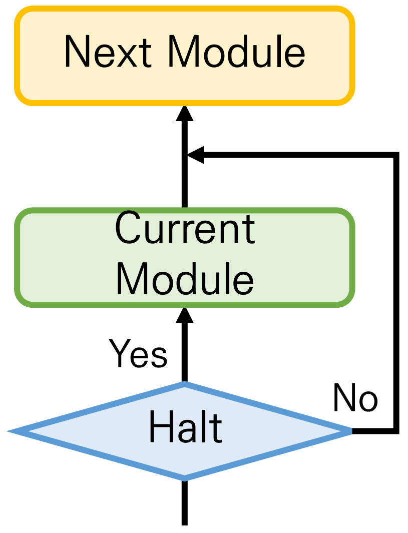
Concept

Connen et al., Visualizing and Measuring the Geometry of BERT, NeurIPS, 2019
[Paper Review] Syntax and Semantics in Language Model Representation (Myeongsup Kim, 2020)

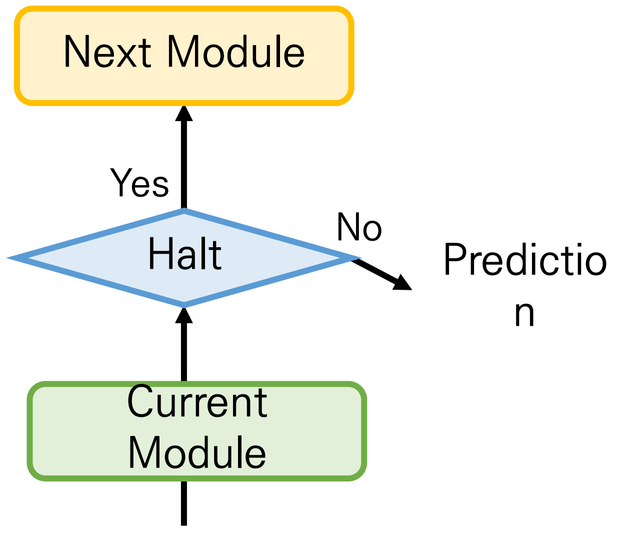
- Three Types of ACT Paradigms
  - Adaptive Computation Time은 다음과 같은 3가지의 분류로 나뉨
    - Dynamic Halting: 조건에 따라 중지 / 재학습
    - Conditional Skipping: 조건에 따라 현재 Module Skip / 현재 Module 학습
    - Early Exit: 조건에 따라 종료 후 Prediction / 다음 Module로 진행



Dynamic Halting



Conditional Skipping



Early Exit

06 Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

Connen et al., Visualizing and Measuring the Geometry of BERT, NeurIPS, 2019
[Paper Review] Syntax and Semantics in Language Model Representation (Myeongsup Kim, 2020)

- Universal Transformer
  - Transformer는 Contextualized Representation을 얻기 위해 전적으로 Self-Attention에 의존함
  - Self-Attention은 병렬화가 가능하다는 장점이 있지만, Global한 Receptive Field를 갖게 됨
    - Sequence에 포함된 모든 Token의 정보를 항상 반영하게 됨
    - Semantic Boundary를 무시하고 Sequence에 포함된 모든 Token의 정보를 반영한다는 한계 존재 (Connen et al., 2019)
    - Recurrent Inductive Bias를 부과할 수 없음
  - 결과적으로 제한된 범위의 Receptive Field를 갖는 CNN과 RNN이 쉽게 풀 수 있었던 일부 Task를 수행하기 어려워 짐
    - e.g. Copying Strings, Logical Inference
  - 이러한 한계를 극복하고자 Parallel-in-Time-Recurrent Self-Attentive Sequence Model인 Universal Transformer를 제안
    - Recurrent Step (Encoder / Decoder)
    - Dynamic Halting

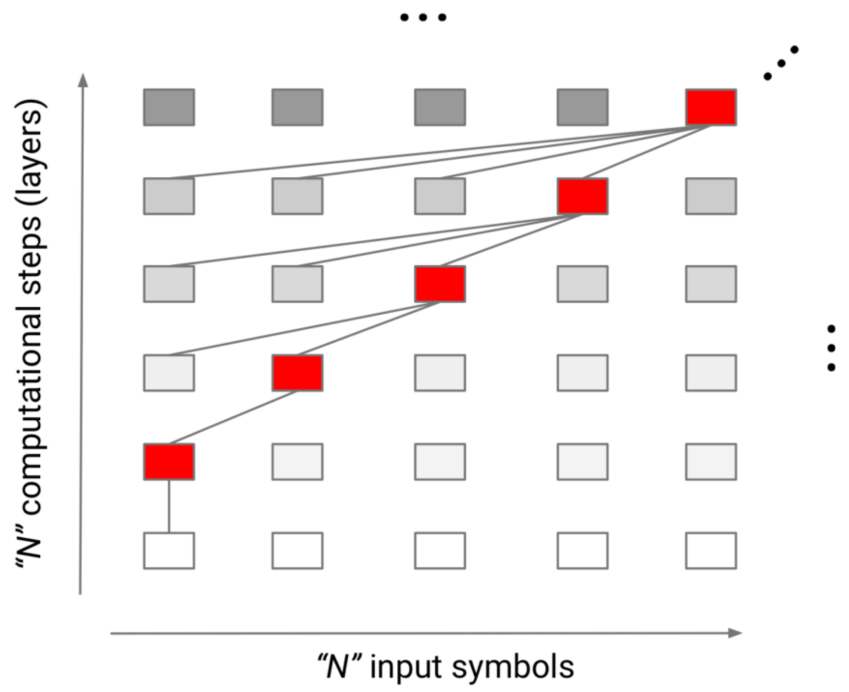
06

Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

Connen et al., Visualizing and Measuring the Geometry of BERT, NeurIPS, 2019
[Paper Review] Syntax and Semantics in Language Model Representation (Myeongsup Kim, 2020)

- **Universal Transformer**
  - Self-Attention은 병렬화가 가능하다는 장점이 있지만, Global한 Receptive Field를 갖게 됨
    - Recurrent Inductive Bias를 부과할 수 없음



Recurrent Inductive Bias

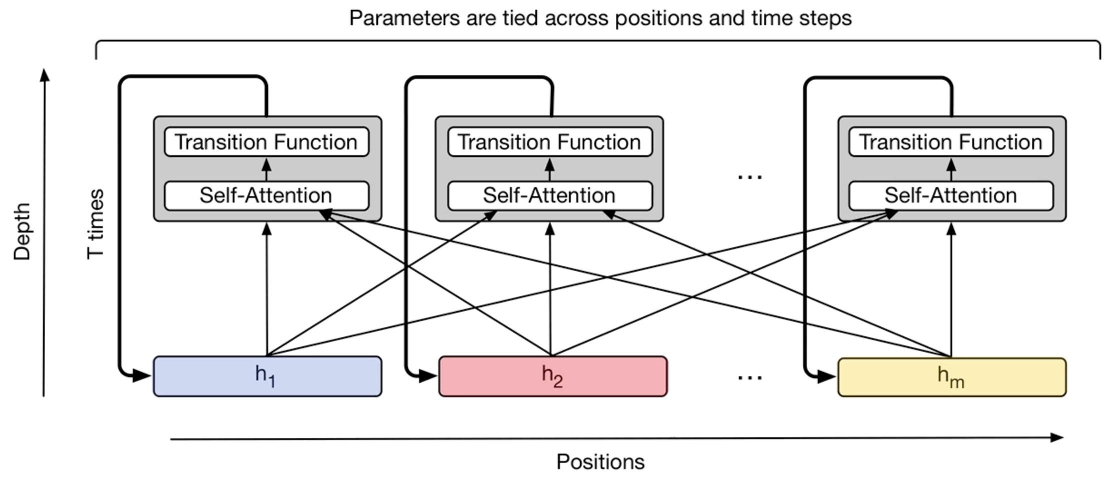
06

Adaptive Computation Time

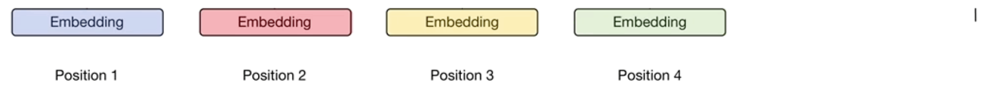
Universal Transformers (ICLR, 2019, Citation: 331)

https://mostafadehghani.com/2019/05/05/universal-transformers/

- **Universal Transformer**
  - Universal Transformer의 Key Idea는 Recurrent Step과 Dynamic Halting
  - Recurrent Step에서는 재귀적으로 Encoder / Decoder를 학습 → Recurrent Inductive Bias 부과
  - Dynamic Halting에서는 가변적으로 특정 Token에 대한 Representation의 학습을 중지 → 선택적으로 Representation에 정보 반영



Recurrent Step

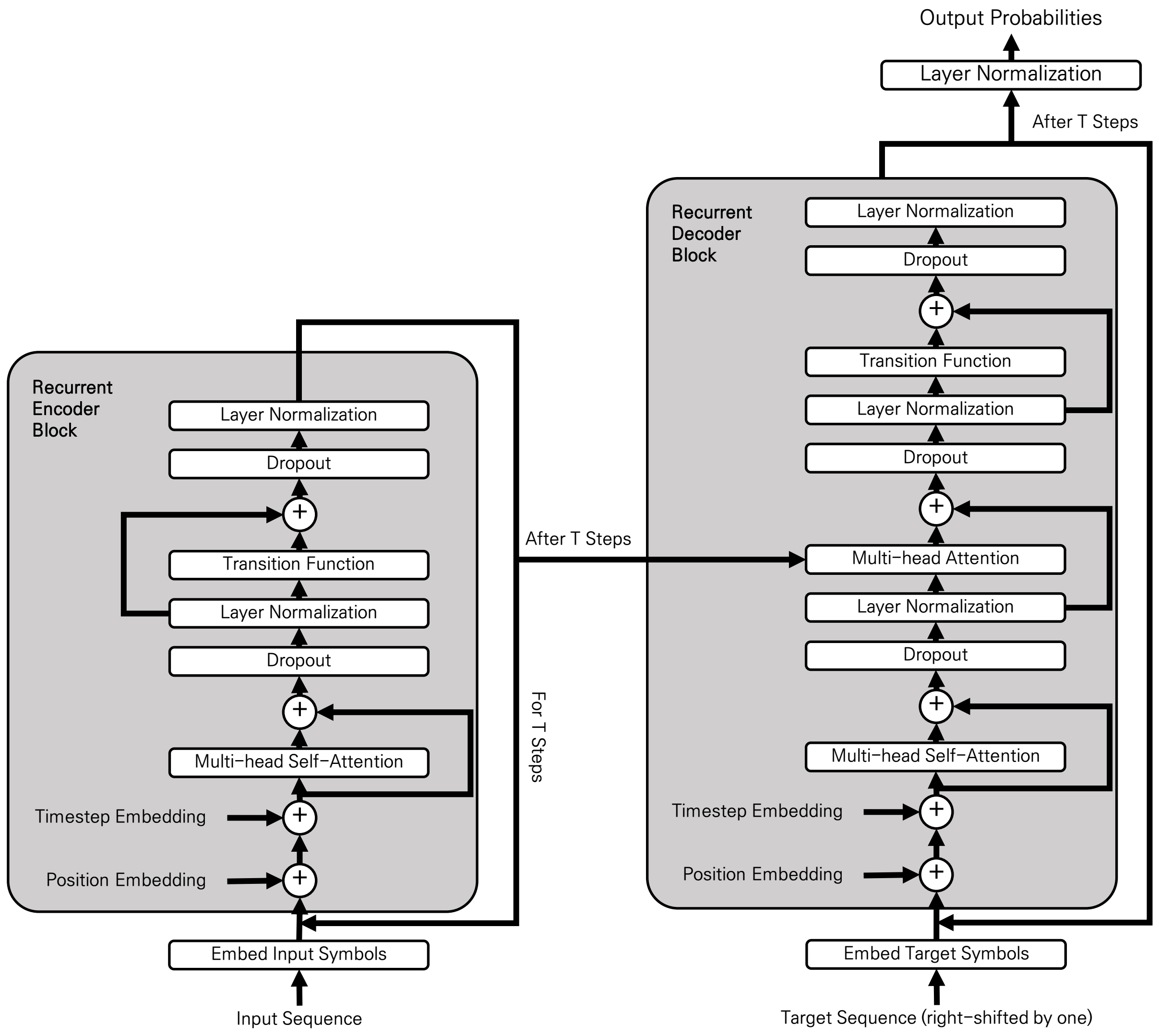


Dynamic Halting

06

Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

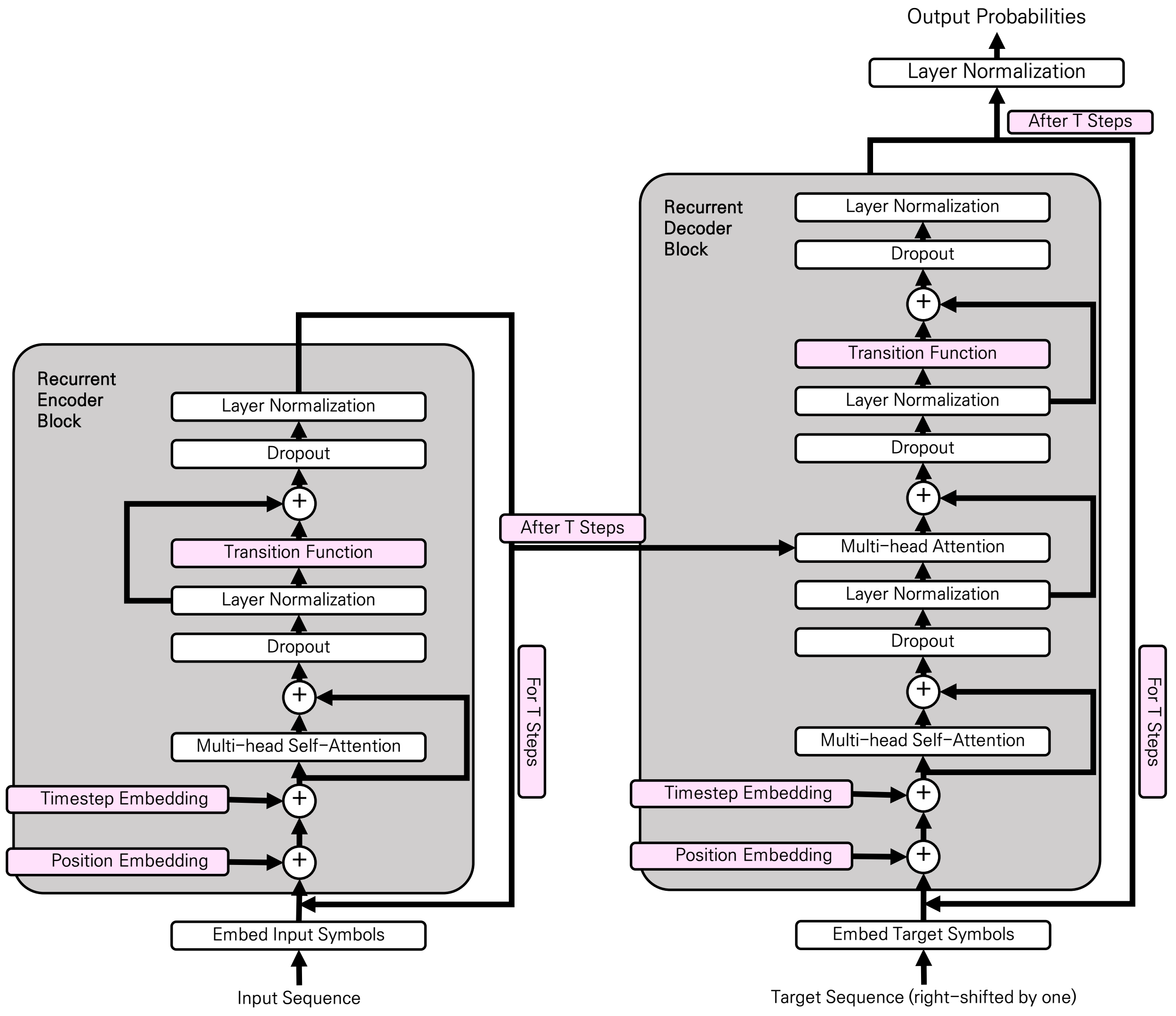


Overall Architecture

06

Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)



Overall Architecture

Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

- **Universal Transformer (Timestep Embedding)**
  - Universal Transformer는 기존 Positional Embedding에 추가적으로 Timestep Embedding 더함
  - Timestep Embedding은 Positional Embedding과 동일한 Sinusoid Embedding을 사용
  - Positional Embedding은 Token의 위치에 따라 다른 값을, Timestep Embedding은 Timestep에 따라 다른 값을 추가
  - Timestep Embedding과 Positional Embedding을 더한 Coordinate Embedding을 최종적으로 사용

$$P_{i,2j}^{t} = \overbrace{\sin\left(i/10000^{2j/d}\right)}^{\text{Positional Embedding}} + \overbrace{\sin\left(t/10000^{2j/d}\right)}^{\text{Timestep Embedding}}$$

$$P_{i,2j+1}^{t} = \cos\left(i/10000^{2j/d}\right) + \cos\left(t/10000^{2j/d}\right)$$

$$sequence\ position\ 1 \leq i \leq m$$

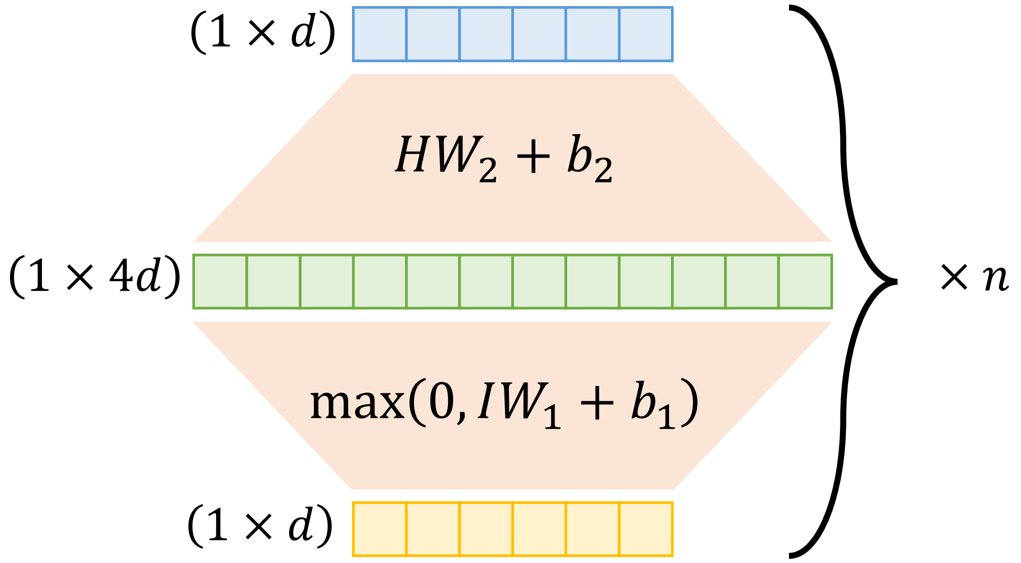
$$embedding\ dimension\ 1 \leq j \leq d$$

$$time\ step\ 1 \leq t \leq T$$

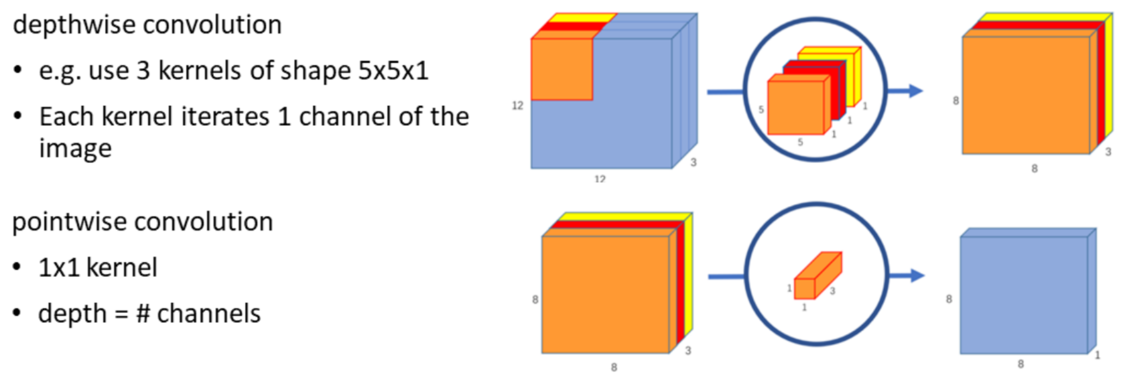
06 Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

- **Universal Transformer (Transition Function)**
  - Transition Function의 경우 기존 Transformer에서 사용된 Position-wise FFN 또는 Separable Convolution을 사용할 수 있음



Position-wise FFNs


depthwise convolution
• e.g. use 3 kernels of shape 5x5x1
• Each kernel iterates 1 channel of the image
pointwise convolution
• 1x1 kernel
• depth = # channels

Separable Convolutions

06

Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

https://glassboxmedicine.com/2019/09/07/universal-transformers/

- **Universal Transformer (Dynamic Halting)**

  Here is a quick summary of how adaptive computation time (ACT) works:

  - At each step we are given:
    - Halting probability and the previous state (initialized as zero)
    - A scalar halt threshold between 0 and 1 (a hyperparameter – i.2. we choose the halt threshold ourselves)
  - First, we compute the new state of each position, using the Universal Transformer
  - Then we compute the "pondering" value using a fully-connected that takes the state down to dimension 1, and applies a sigmoid activation to make the output a probability-like value between 0 and 1. This is pondering value. The "pondering" value is the model's estimation of how much additional computation is needed for each of the input symbols
  - We decide to halt for any positions that cross the halt threshold:
    - Just halted at this step: (halting probability + pondering) > halt threshold
    - Still running: (halting probability + pondering) ≤ halt threshold
  - For position that are still running, update the halting probability: halting probability += pondering
  - Update the State of the other positions until the model halts all positions or reaches a predefined max number of steps

06

Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

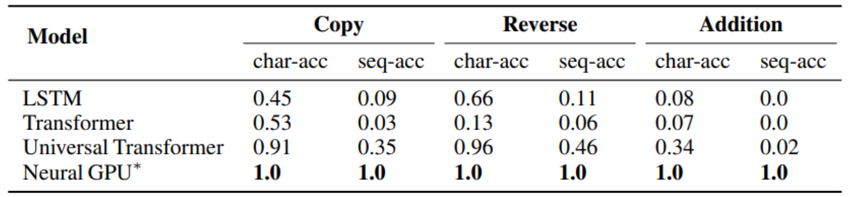
https://glassboxmedicine.com/2019/09/07/universal-transformers/

- **Universal Transformer (Dynamic Halting)**
    - 매 Step마다 아래의 값이 주어짐:
        - Halting Probability와 이전 State (0으로 초기화)
        - 0과 1사이의 값을 갖는 Halt Threshold (Hyper parameter)
    - Universal Transformer를 이용해 각 Position마다 New State를 계산
    - Pondering Value를 계산, 1차원으로 변환하는 Fully-connected Layer와 Sigmoid 사용, 0과 1사이의 값을 가짐
    - Pondering Value는 각 Token (Symbol)에 대해 추가적인 연산이 필요할지에 대한 Model의 추정값
    - 각 Position마다 Halt Threshold를 사용하여 다음 중 하나를 결정:
        - 이번 Step에서는 학습 중지 (Halting Probability + Pondering) > Halt Threshold
        - 계속 학습 (Halting Probability + Pondering) ≤ Halt Threshold
    - 각 Position이 학습 중이면 Halt Probability를 Update: Halting Probability += Pondering
    - Model이 모든 Position에 대해 학습을 중지하거나 Max Step에 도달할 때 까지 State를 Update

06 Adaptive Computation Time

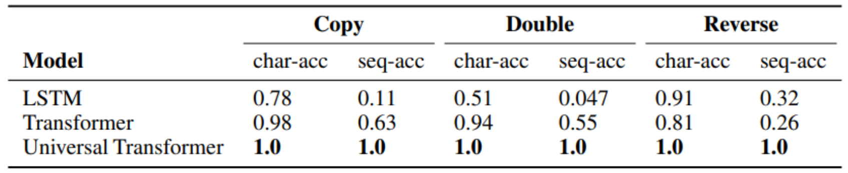
Universal Transformers (ICLR, 2019, Citation: 331)

- Universal Transformer
  - Vanilla Transformer가 잘 수행하지 못했던 다수의 Task에서 높은 성능을 보임
  - Neural Machine Translation Task에서 Transformer를 상회하는 성능을 기록



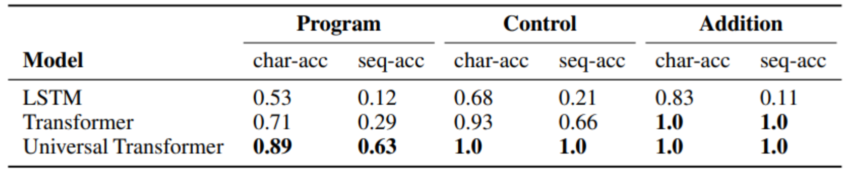
Model	Copy		Reverse		Addition	
	char-acc	seq-acc	char-acc	seq-acc	char-acc	seq-acc
LSTM	0.45	0.09	0.66	0.11	0.08	0.0
Transformer	0.53	0.03	0.13	0.06	0.07	0.0
Universal Transformer	0.91	0.35	0.96	0.46	0.34	0.02
Neural GPU*	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

Table 4: Accuracy (higher better) on the algorithmic tasks. *Note that the Neural GPU was trained with a special curriculum to obtain the perfect result, while other models are trained without any curriculum.



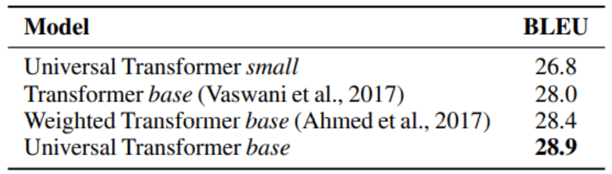
Model	Copy		Double		Reverse	
	char-acc	seq-acc	char-acc	seq-acc	char-acc	seq-acc
LSTM	0.78	0.11	0.51	0.047	0.91	0.32
Transformer	0.98	0.63	0.94	0.55	0.81	0.26
Universal Transformer	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

Table 5: Character-level (*char-acc*) and sequence-level accuracy (*seq-acc*) results on the Memorization LTE tasks, with maximum length of 55.



Model	Program		Control		Addition	
	char-acc	seq-acc	char-acc	seq-acc	char-acc	seq-acc
LSTM	0.53	0.12	0.68	0.21	0.83	0.11
Transformer	0.71	0.29	0.93	0.66	**1.0**	**1.0**
Universal Transformer	**0.89**	**0.63**	**1.0**	**1.0**	**1.0**	**1.0**

Table 6: Character-level (*char-acc*) and sequence-level accuracy (*seq-acc*) results on the Program Evaluation LTE tasks with maximum nesting of 2 and length of 5.



Model	BLEU
Universal Transformer *small*	26.8
Transformer *base* (Vaswani et al., 2017)	28.0
Weighted Transformer *base* (Ahmed et al., 2017)	28.4
Universal Transformer *base*	**28.9**

Table 7: Machine translation results on the WMT14 En-De translation task trained on 8xP100 GPUs in comparable training setups. All *base* results have the same number of parameters.

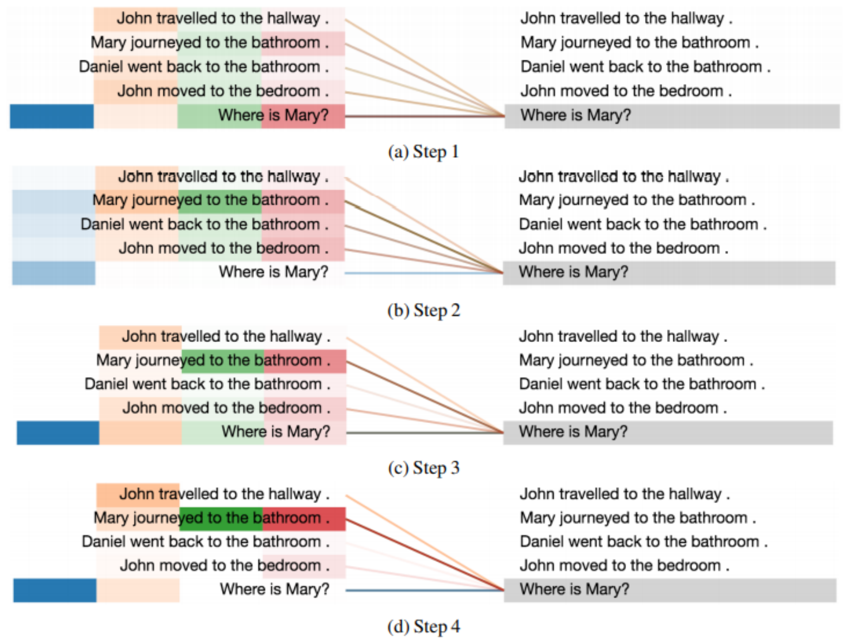
06

Adaptive Computation Time

Universal Transformers (ICLR, 2019, Citation: 331)

- Universal Transformer
  - Reasoning Task에서 좋은 성능을 보임

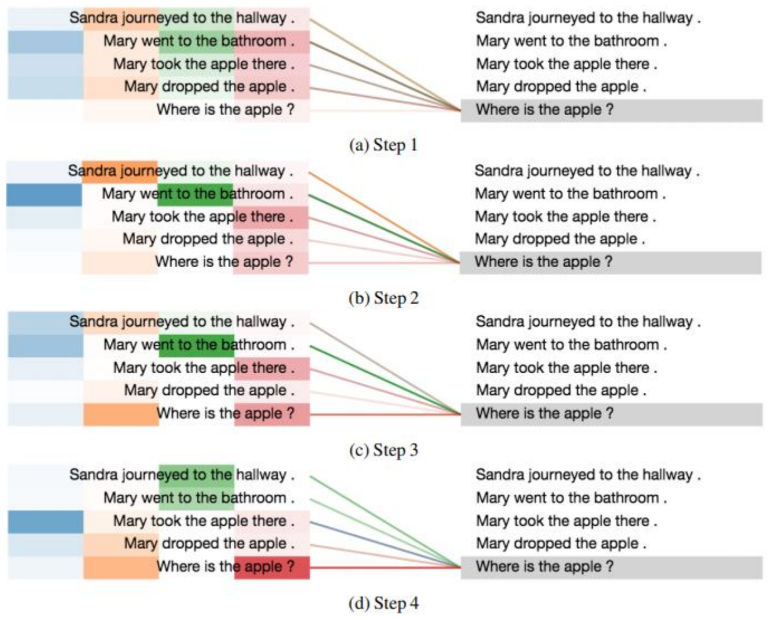
```
An example from tasks 1:   (requiring one supportive fact to solve)
Story:
                            John travelled to the hallway.
                            Mary journeyed to the bathroom.
                            Daniel went back to the bathroom.
                            John moved to the bedroom
Question:
                            Where is Mary?
Model's output:
                            bathroom
```



John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
(a) Step 1
(b) Step 2
(c) Step 3
(d) Step 4

Figure 5: Visualization of the attention distributions, when encoding the question: *"Where is Mary?"*.

```
An example from tasks 2:   (requiring two supportive facts to solve)
Story:
                            Sandra journeyed to the hallway.
                            Mary went to the bathroom.
                            Mary took the apple there.
                            Mary dropped the apple.
Question:
                            Where is the apple?
Model's output:
                            bathroom
```



Sandra journeyed to the hallway .
Mary went to the bathroom .
Mary took the apple there .
Mary dropped the apple .
Where is the apple ?
(a) Step 1
(b) Step 2
(c) Step 3
(d) Step 4

Figure 6: Visualization of the attention distributions, when encoding the question: *"Where is the apple?"*.

06

Adaptive Computation Time

DeeBERT (ACL, 2020, Citation: 49)

Connen et al., Visualizing and Measuring the Geometry of BERT, NeurIPS, 2019
Hewitt and Manning, A Structural Probe for Finding Syntax in Word Representaiton, NAACL, 2019
[Paper Review] Syntax and Semantics in Language Model Representation (Myeongsup Kim, 2020)

- DeeBERT
  - BERT는 좋은 성능을 보이고 있지만, 전체 Model을 이용하기 때문에 Inference 속도가 느리다는 단점이 존재
  - Pre-trained BERT의 각 Layer마다 표상하는 정보가 서로 다름 (Hewitt and Manning, 2019; Connen et al., 2019)
  - 전체 Model을 사용하지 않고 Inference가 가능하도록 Early Exiting을 적용

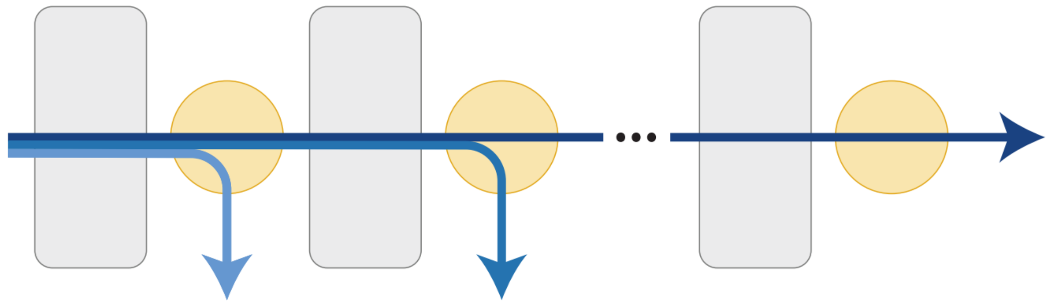


Figure 1: DeeBERT model overview. Grey blocks are transformer layers, orange circles are classification layers (off-ramps), and blue arrows represent inference samples exiting at different layers.

06

Adaptive Computation Time

DeeBERT (ACL, 2020, Citation: 49)

- DeeBERT
  - Loss를 이용해 Embedding Layer, Transformer Layer, Classifier를 Fine-tuning
  - Fine-tuning된 모든 Layer Parameter를 Freeze하고, Classifier만 Update
  - 낮은 순서로 각 Layer Representation와 Classifier를 사용해 Entropy를 계산, 조건을 만족하면 해당 Representation으로 Inference

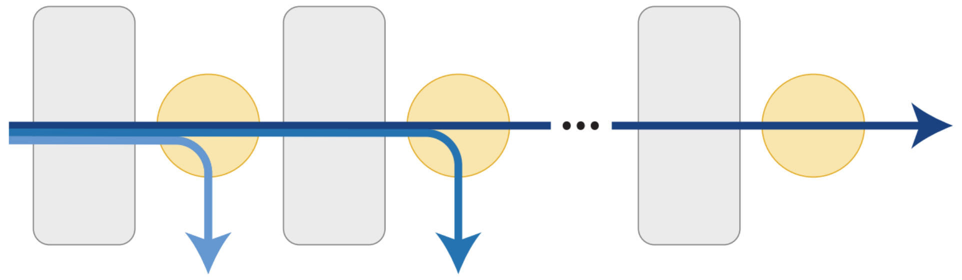


Figure 1: DeeBERT model overview. Grey blocks are transformer layers, orange circles are classification layers (off-ramps), and blue arrows represent inference samples exiting at different layers.

Algorithm 1 DeeBERT Inference (Input: x)

```
**for** $i = 1$ to $n$ **do**
    $z_i = f_i(x;\theta)$
    **if** entropy$(z_i) < S$ **then**
        **return** $z_i$
    **end if**
**end for**
**return** $z_n$
```

Adaptive Computation Time

DeeBERT (ACL, 2020, Citation: 49)

- **DeeBERT**
  - 대체적으로 Exit Layer가 높아질수록 성능이 증가하는 경향이 발견됨, Inference Time과 Performance 사이의 Trade-off 존재

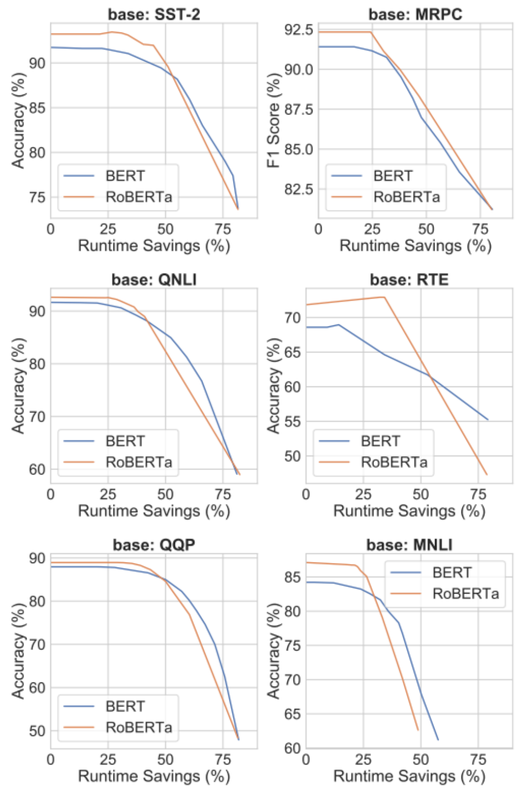


Figure 2: DeeBERT quality and efficiency trade-offs for BERT-base and RoBERTa-base models.

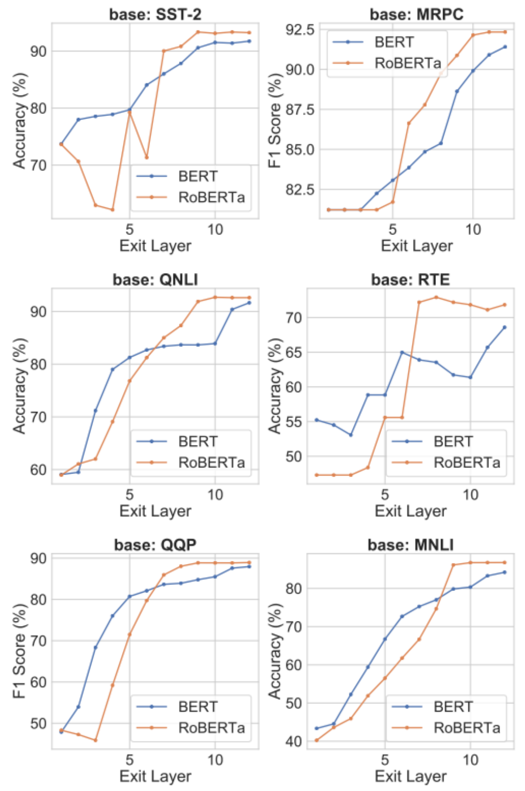


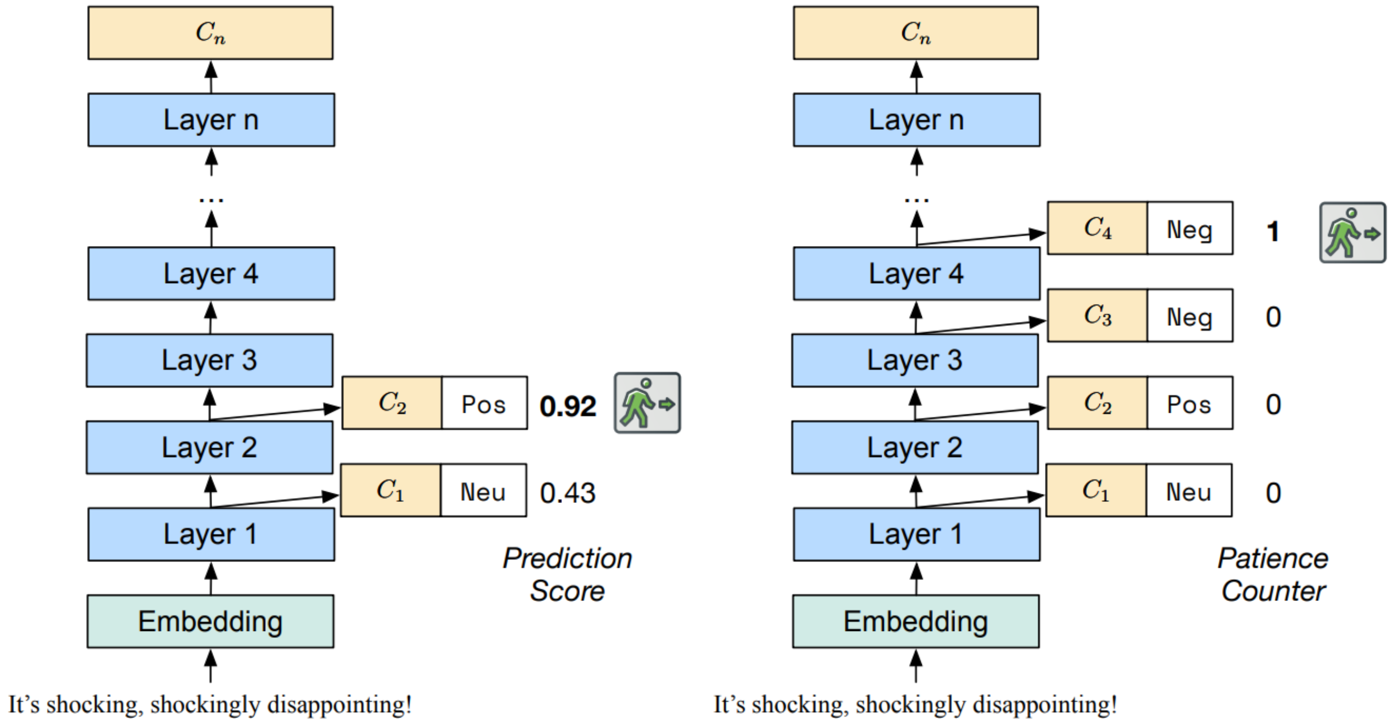
Figure 4: Accuracy of each off-ramp for BERT-base and RoBERTa-base.

06

Adaptive Computation Time

PABEE (NeurIPS, 2020, Citation: 29)

- PABEE
  - Patience-based Early Exit (PABEE)는 Pre-trained LM의 Robustness와 Efficiency를 향상시킬 수 있음
  - Plug-and-Play 방식으로 작동 가능, 각 Layer의 Representation과 Classifier를 이용한 Early Exit 수행



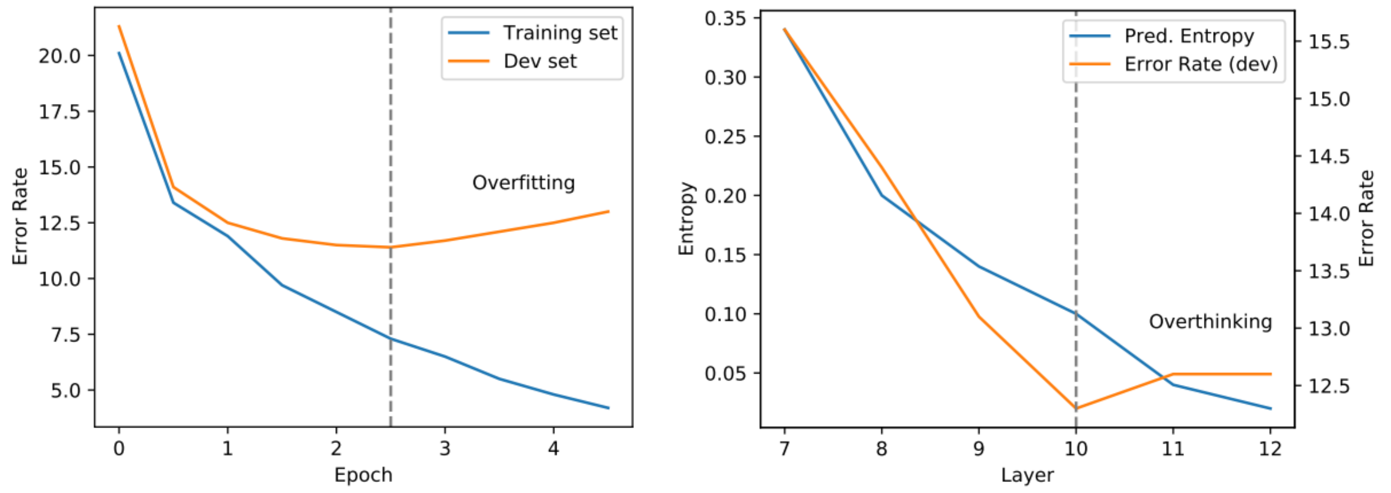
(a) Shallow-Deep Net [5] (b) Patience-based Early Exit (PABEE)

06

Adaptive Computation Time

PABEE (NeurIPS, 2020, Citation: 29)

- PABEE
  - Training에서의 Overfitting과 비슷하게, Inference 시에 Overthinking 개념을 제시
  - LM의 각 Layer Representation을 사용했을 때 Layer가 깊어짐에 따라 Entropy는 감소하지만, Error Rate는 증가하는 상황이 발견됨



(a) Overfitting in training

(b) Overthinking in inference

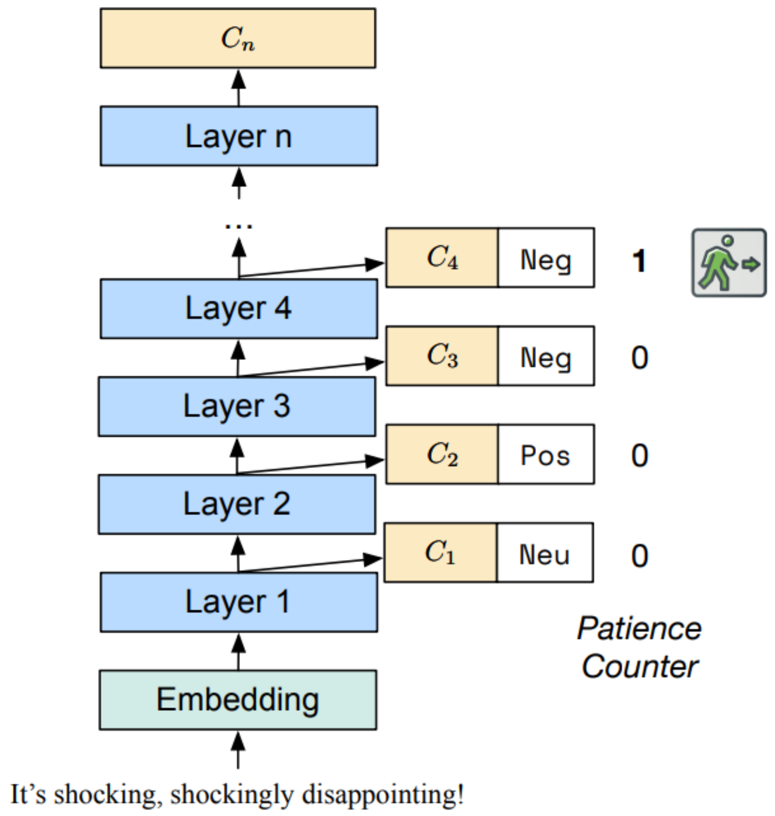
Figure 2: Analogy between overfitting in training and overthinking in inference. **(a)** In training, the error rate keeps going down on the training set but goes up later on the development set. **(b)** We insert a classifier after every layer. Similarly, the predicted entropy keeps dropping when more layers are added to inference but the error rate goes up after 10 layers. The results are obtained with ALBERT-base on MRPC.

06

Adaptive Computation Time

PABEE (NeurIPS, 2020, Citation: 29)

- PABEE (Inference)
  - Language Model 각 Layer의 Representation과 Classifier를 이용해서 Inference를 수행
  - Threshold인 t번 이상 연속적인 Layer에서 같은 Label로 예측할 경우, 다음 Layer로 진행하지 않고 Early Stop



$$L_i : Layers$$

$$\boldsymbol{h}_i = L_i(\boldsymbol{h}_{i-1})$$

$$\boldsymbol{h}_0 = Embedding$$

$$\boldsymbol{y}_i : Predicted\ Label\ of\ h_i$$

$$cnt_i = \begin{cases} cnt_{i-1} + 1 & \arg\max(\boldsymbol{y}_i) = \arg\max(\boldsymbol{y}_{i-1}) \\ 0 & \arg\max(\boldsymbol{y}_i) \neq \arg\max(\boldsymbol{y}_{i-1}) \vee i = 0 \end{cases}$$

$$if\ cnt_i = t \ \rightarrow stop\ Inference$$

06 Adaptive Computation Time

PABEE (NeurIPS, 2020, Citation: 29)

- PABEE (Training)
  - PABEE는 Language Model 각 Layer의 Representation과 Classifier를 이용해서 Inference를 수행
  - Classifier가 각 Layer의 Representation을 이용해서 예측을 수행할 수 있도록 학습이 필요
  - 각 Layer의 Representation으로 계산된 Cross Entropy Loss의 평균을 이용하여 학습

$$\mathcal{L}_i = -\sum_{z \in Z} [\mathbb{1}[\boldsymbol{y}_i = z] \cdot \log P(\boldsymbol{y}_i = z | \boldsymbol{h}_i)]$$

$$\mathcal{L} = \frac{\sum_{j=1}^{n} j \cdot \mathcal{L}_j}{\sum_{j=1}^{n} j}$$

06

Adaptive Computation Time

PABEE (NeurIPS, 2020, Citation: 29)

- PABEE
  - GLUE Benchmark에서 ALBERT를 Backbone으로 사용했을 때 성능과 속도에서 모두 우수함을 확인

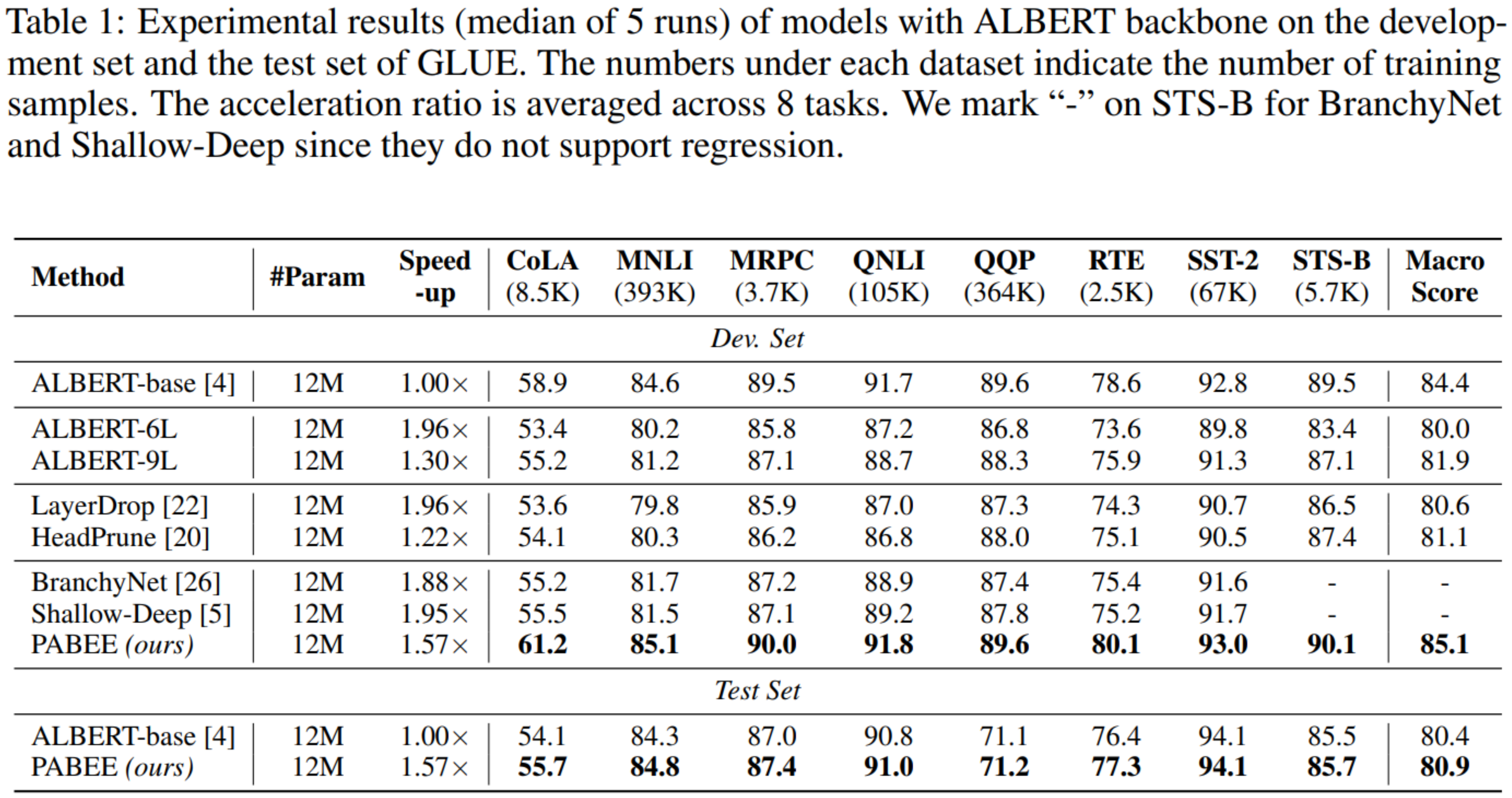


Table 1: Experimental results (median of 5 runs) of models with ALBERT backbone on the development set and the test set of GLUE. The numbers under each dataset indicate the number of training samples. The acceleration ratio is averaged across 8 tasks. We mark "-" on STS-B for BranchyNet and Shallow-Deep since they do not support regression.

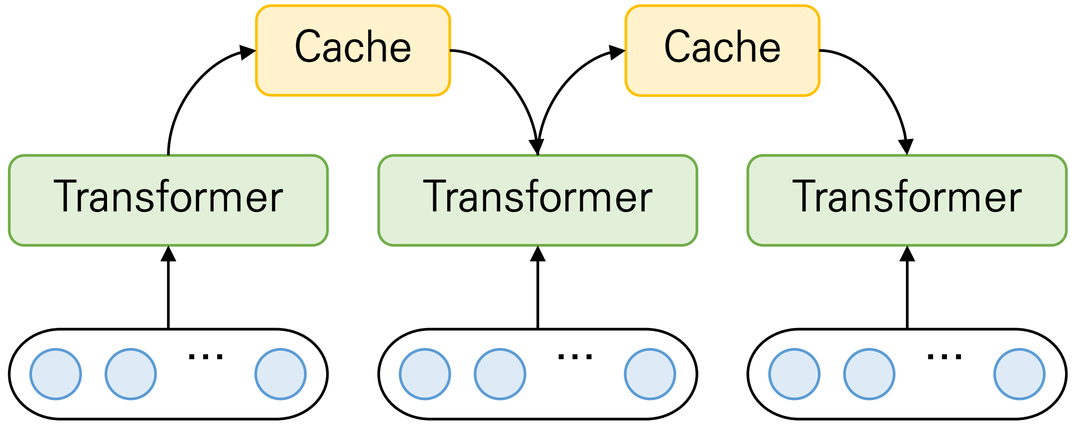
Method	**#Param**	**Speed -up**	**CoLA** (8.5K)	**MNLI** (393K)	**MRPC** (3.7K)	**QNLI** (105K)	**QQP** (364K)	**RTE** (2.5K)	**SST-2** (67K)	**STS-B** (5.7K)	**Macro Score**
						Dev. Set					
ALBERT-base [4]	12M	1.00×	58.9	84.6	89.5	91.7	89.6	78.6	92.8	89.5	84.4
ALBERT-6L	12M	1.96×	53.4	80.2	85.8	87.2	86.8	73.6	89.8	83.4	80.0
ALBERT-9L	12M	1.30×	55.2	81.2	87.1	88.7	88.3	75.9	91.3	87.1	81.9
LayerDrop [22]	12M	1.96×	53.6	79.8	85.9	87.0	87.3	74.3	90.7	86.5	80.6
HeadPrune [20]	12M	1.22×	54.1	80.3	86.2	86.8	88.0	75.1	90.5	87.4	81.1
BranchyNet [26]	12M	1.88×	55.2	81.7	87.2	88.9	87.4	75.4	91.6	-	-
Shallow-Deep [5]	12M	1.95×	55.5	81.5	87.1	89.2	87.8	75.2	91.7	-	-
PABEE *(ours)*	12M	1.57×	**61.2**	**85.1**	**90.0**	**91.8**	**89.6**	**80.1**	**93.0**	**90.1**	**85.1**
						Test Set					
ALBERT-base [4]	12M	1.00×	54.1	84.3	87.0	90.8	71.1	76.4	94.1	85.5	80.4
PABEE *(ours)*	12M	1.57×	**55.7**	**84.8**	**87.4**	**91.0**	**71.2**	**77.3**	**94.1**	**85.7**	**80.9**

06

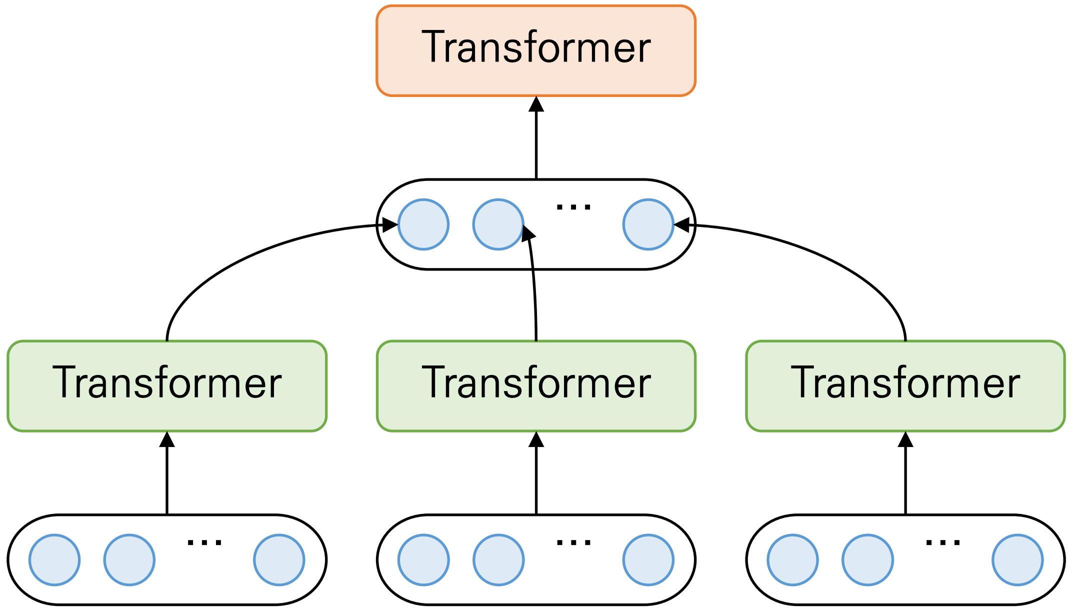
Divide-and-Conquer Strategies

Concept

- **Divide-and-Conquer Strategies**
  - Long Sequence / Richer Representation을 위해 Recurrent 또는 Hierarchical 구조를 사용한 Transformers



Recurrent Transformer



Hierarchical Transformer

06 Divide-and-Conquer Strategies

Hi-Transformer (ACL, 2021, Citation: 1)

- Hi-Transformer
  - Transformer는 Text Modeling에 효과적이지만, Quadratic Complexity로 인해 Long Text를 처리하기에 어려움이 있음
  - 이러한 문제를 해결하기 위해 Hierarchical Interactive Transformer (Hi-Transformer)를 제안
  - Global Document를 Modeling하며 효율적인 Complexity를 유지

06

Divide-and-Conquer Strategies

Hi-Transformer (ACL, 2021, Citation: 1)

- **Hi-Transformer**
  - 각 Sentence를 구성하는 Word를 Sentence Transformer에 입력, Sequence의 마지막에 [CLS] Token 위치
  - Sentence Positional Embedding을 더한 [CLS]를 Document Transformer에 입력, [CLS]와 Sentence Representation을 Pooling

Hi-Transformer Complexity

$$O(M \cdot K^2 \cdot d)$$

Transformer Complexity

$$O(M^2 \cdot K^2 \cdot d)$$

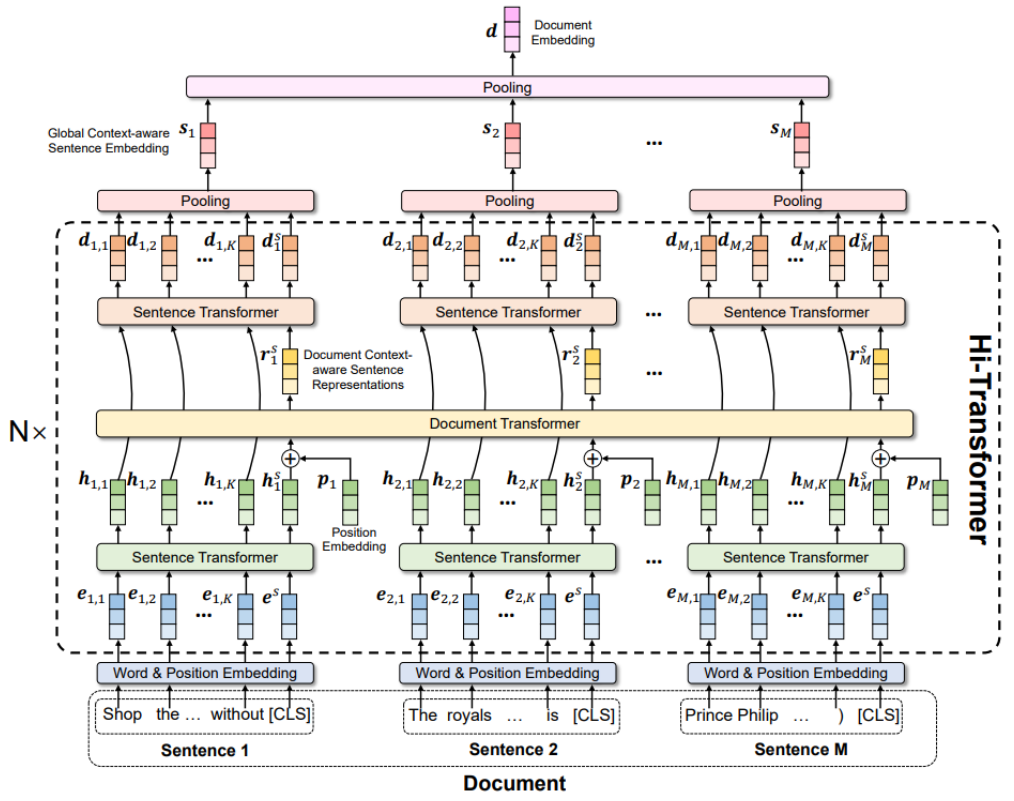
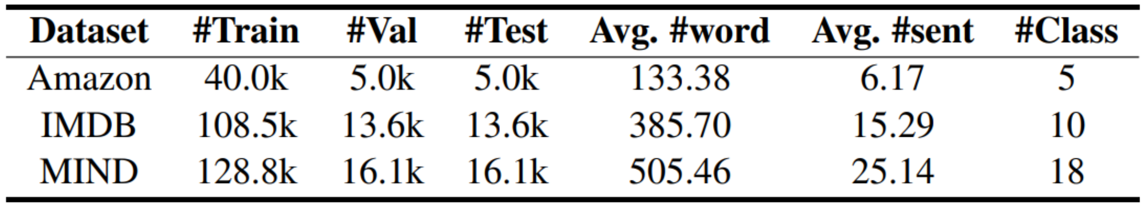


Figure 1: The architecture of *Hi-Transformer*.

06 Divide-and-Conquer Strategies

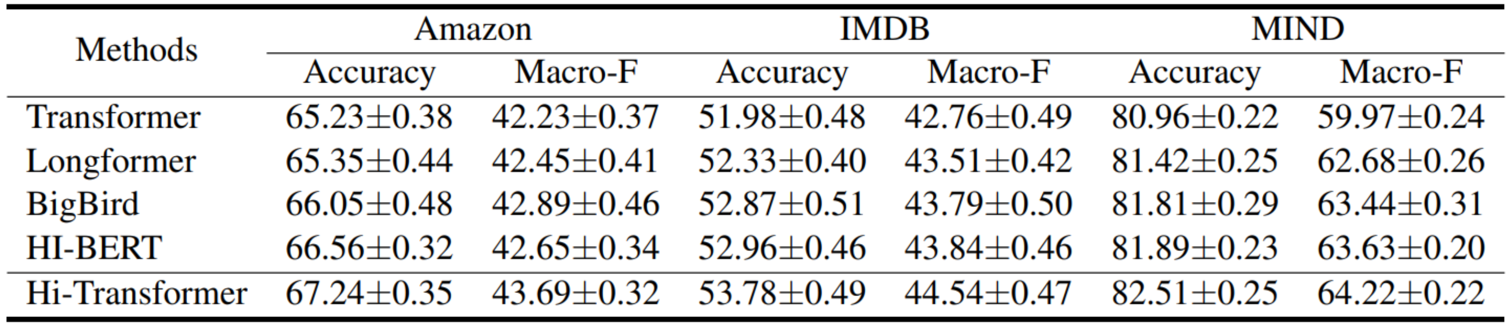
Hi-Transformer (ACL, 2021, Citation: 1)

- Hi-Transformer
  - Baseline Model보다 우수한 성능을 확인
  - Text Length가 길어질수록 성능이 향상, Long Text 처리에 효과적



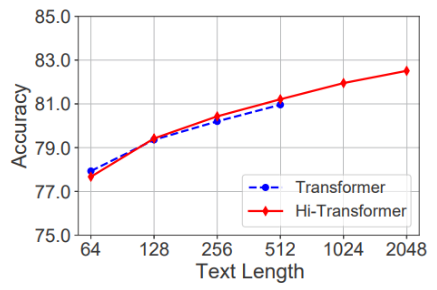
Dataset	#Train	#Val	#Test	Avg. #word	Avg. #sent	#Class
Amazon	40.0k	5.0k	5.0k	133.38	6.17	5
IMDB	108.5k	13.6k	13.6k	385.70	15.29	10
MIND	128.8k	16.1k	16.1k	505.46	25.14	18

Table 1: Statistics of datasets.

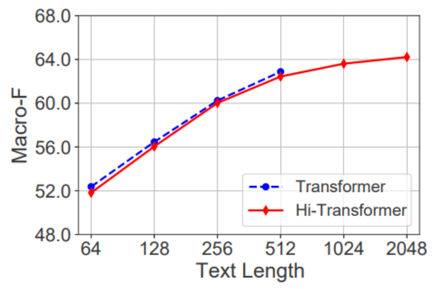


Methods	Amazon		IMDB		MIND	
	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F
Transformer	65.23±0.38	42.23±0.37	51.98±0.48	42.76±0.49	80.96±0.22	59.97±0.24
Longformer	65.35±0.44	42.45±0.41	52.33±0.40	43.51±0.42	81.42±0.25	62.68±0.26
BigBird	66.05±0.48	42.89±0.46	52.87±0.51	43.79±0.50	81.81±0.29	63.44±0.31
HI-BERT	66.56±0.32	42.65±0.34	52.96±0.46	43.84±0.46	81.89±0.23	63.63±0.20
Hi-Transformer	67.24±0.35	43.69±0.32	53.78±0.49	44.54±0.47	82.51±0.25	64.22±0.22

Table 2: The results of different methods on different datasets.



(a) Accuracy.



(b) Macro-F.

Result on MIND Dataset

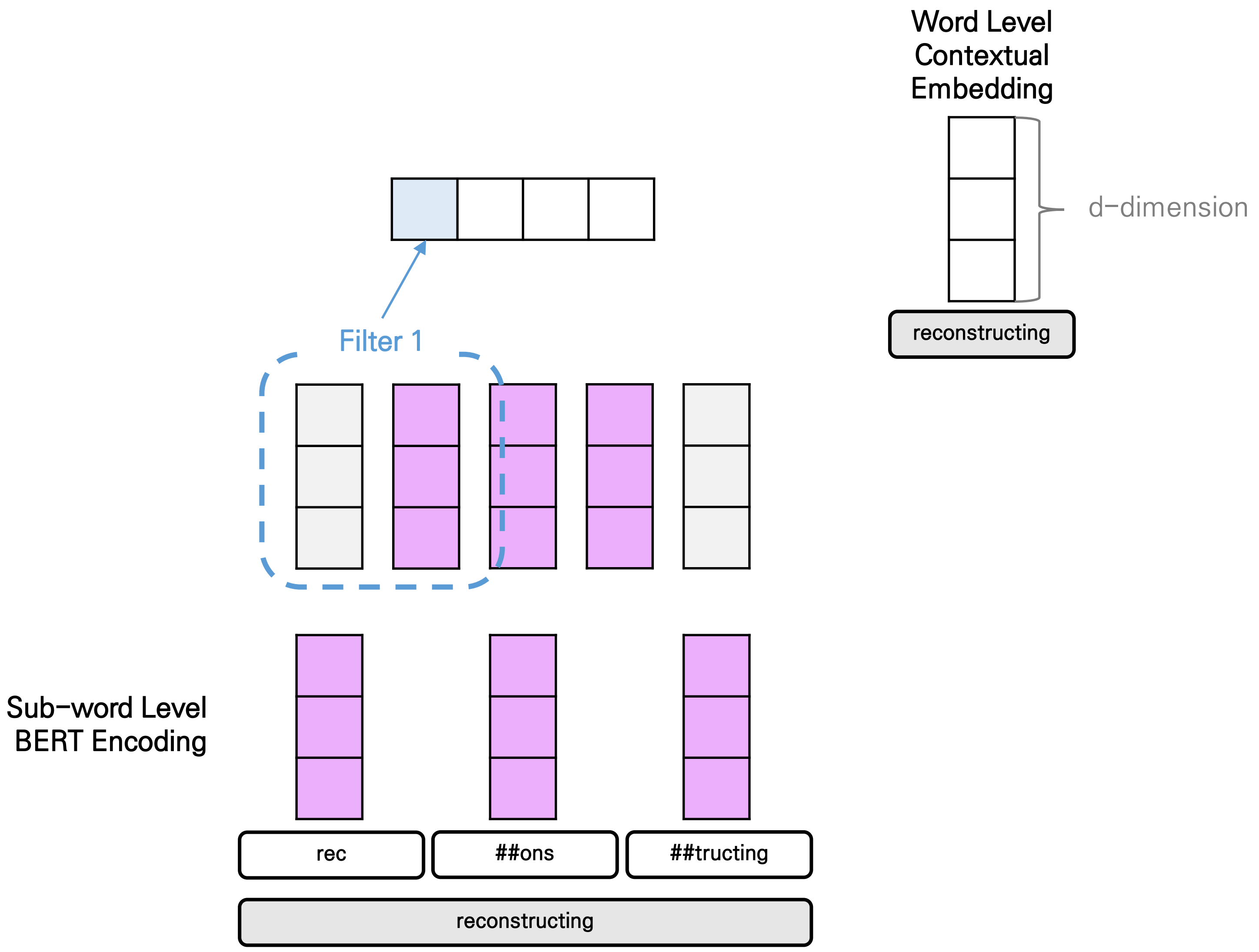
Thank You

00 Appendix 1

1D Convolution

From [Paper Review] Semantics-aware BERT for Language Understanding (Myeongsup Kim, 2020)

⟨1D Convolution⟩

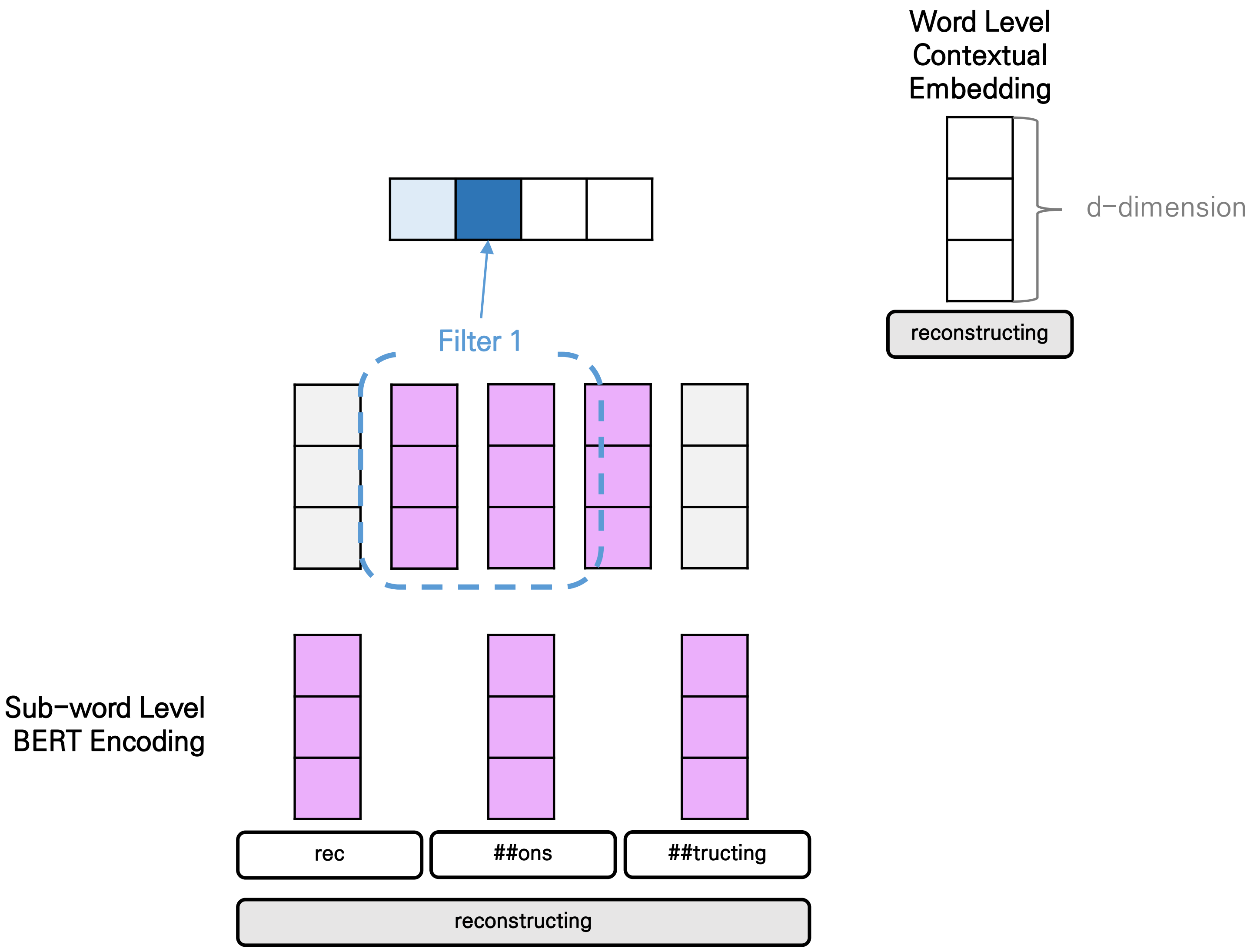


Appendix 1

1D Convolution

From [Paper Review] Semantics-aware BERT for Language Understanding (Myeongsup Kim, 2020)

〈1D Convolution〉



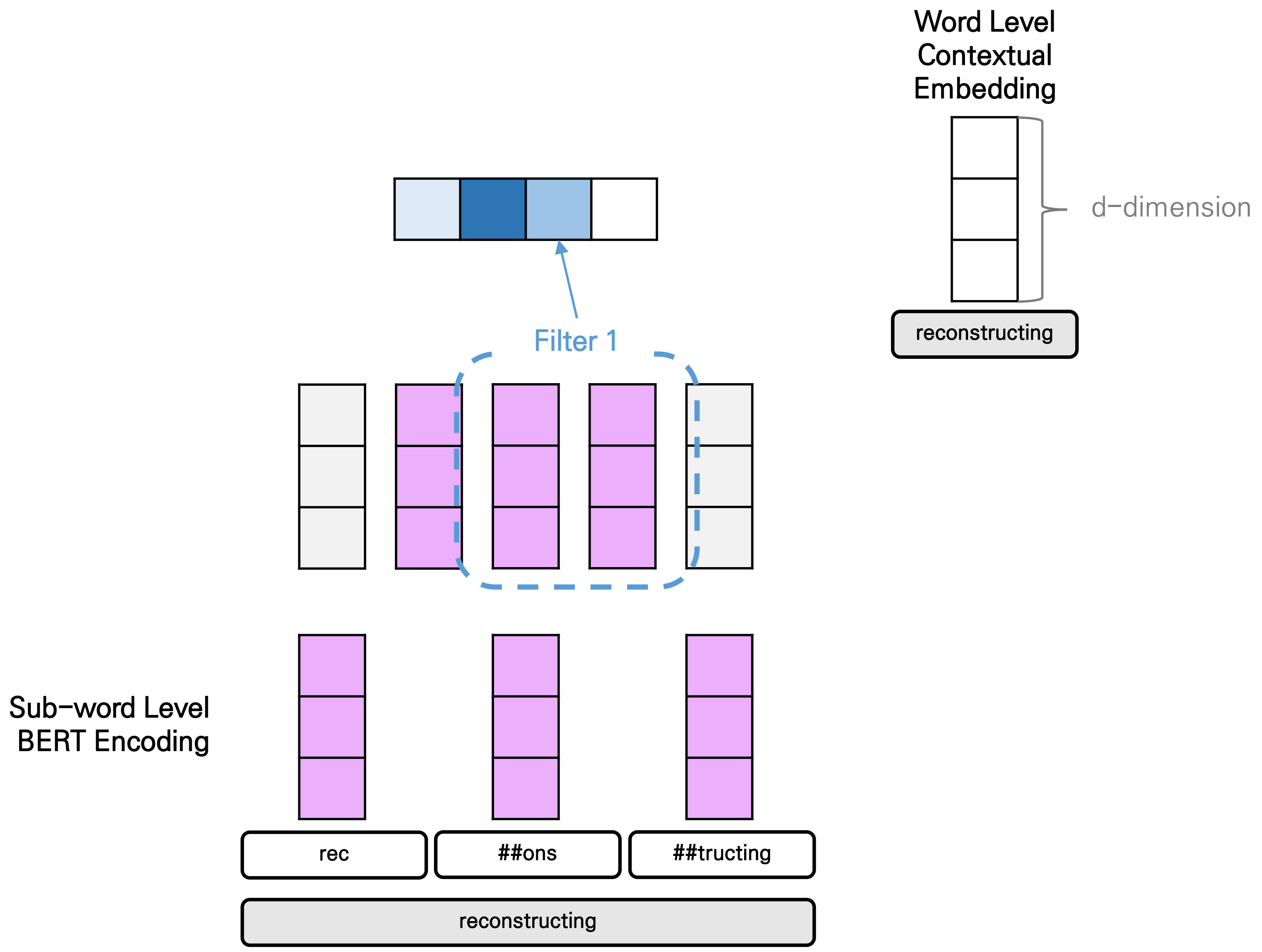
00

Appendix 1

1D Convolution

From [Paper Review] Semantics-aware BERT for Language Understanding (Myeongsup Kim, 2020)

〈1D Convolution〉



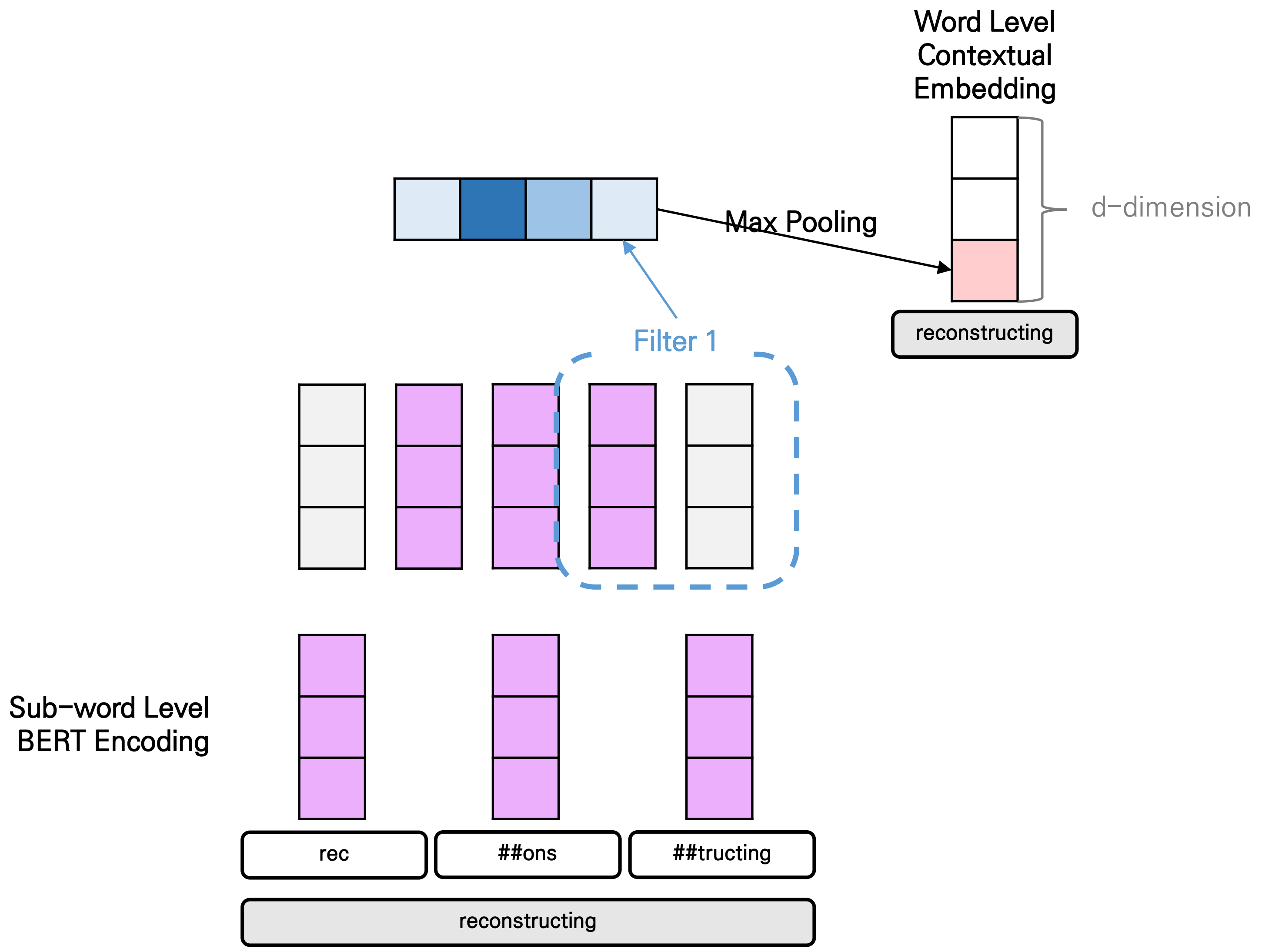
00

Appendix 1

1D Convolution

From [Paper Review] Semantics-aware BERT for Language Understanding (Myeongsup Kim, 2020)

⟨1D Convolution⟩



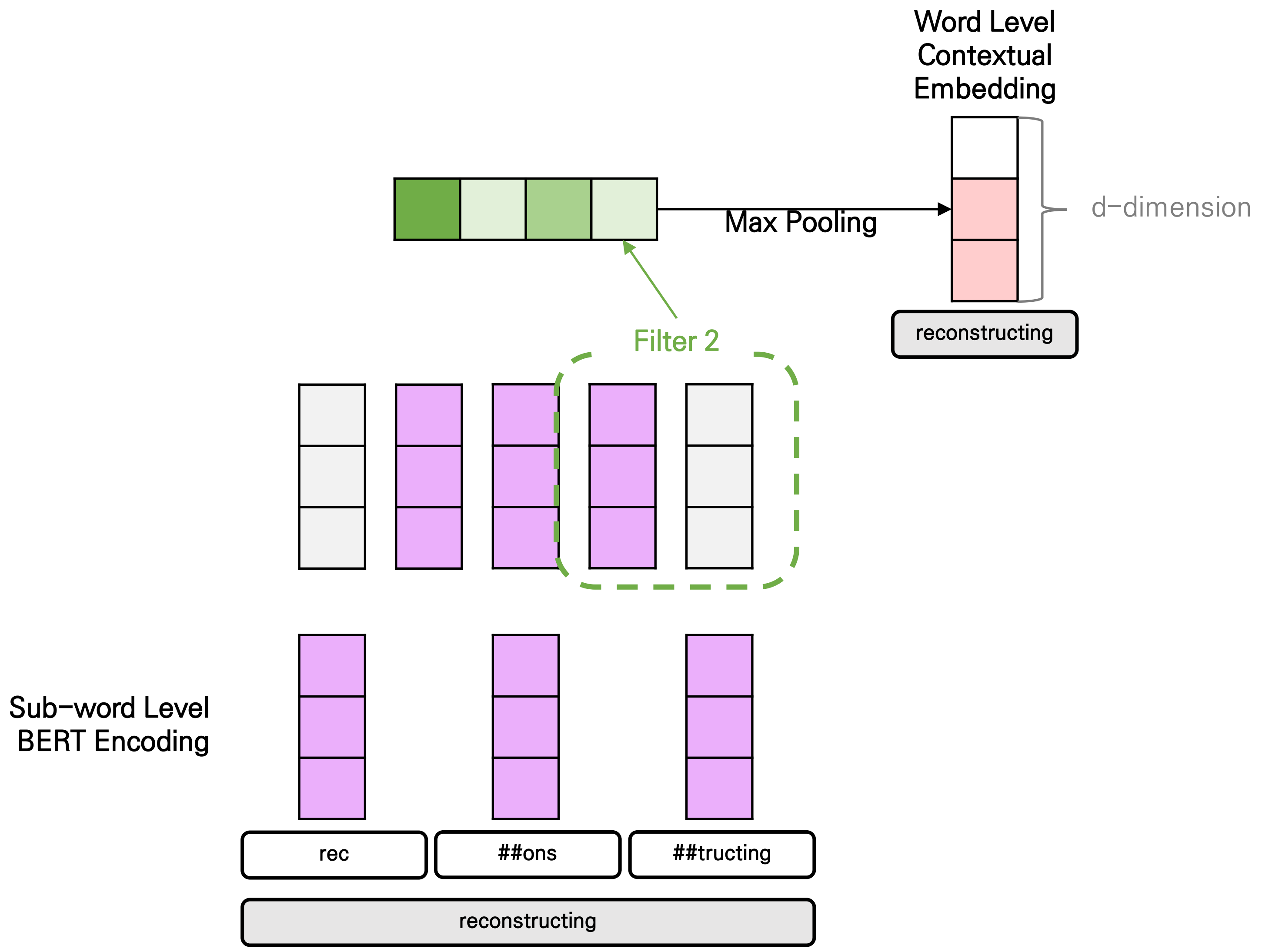
00

Appendix 1

1D Convolution

From [Paper Review] Semantics-aware BERT for Language Understanding (Myeongsup Kim, 2020)

〈1D Convolution〉



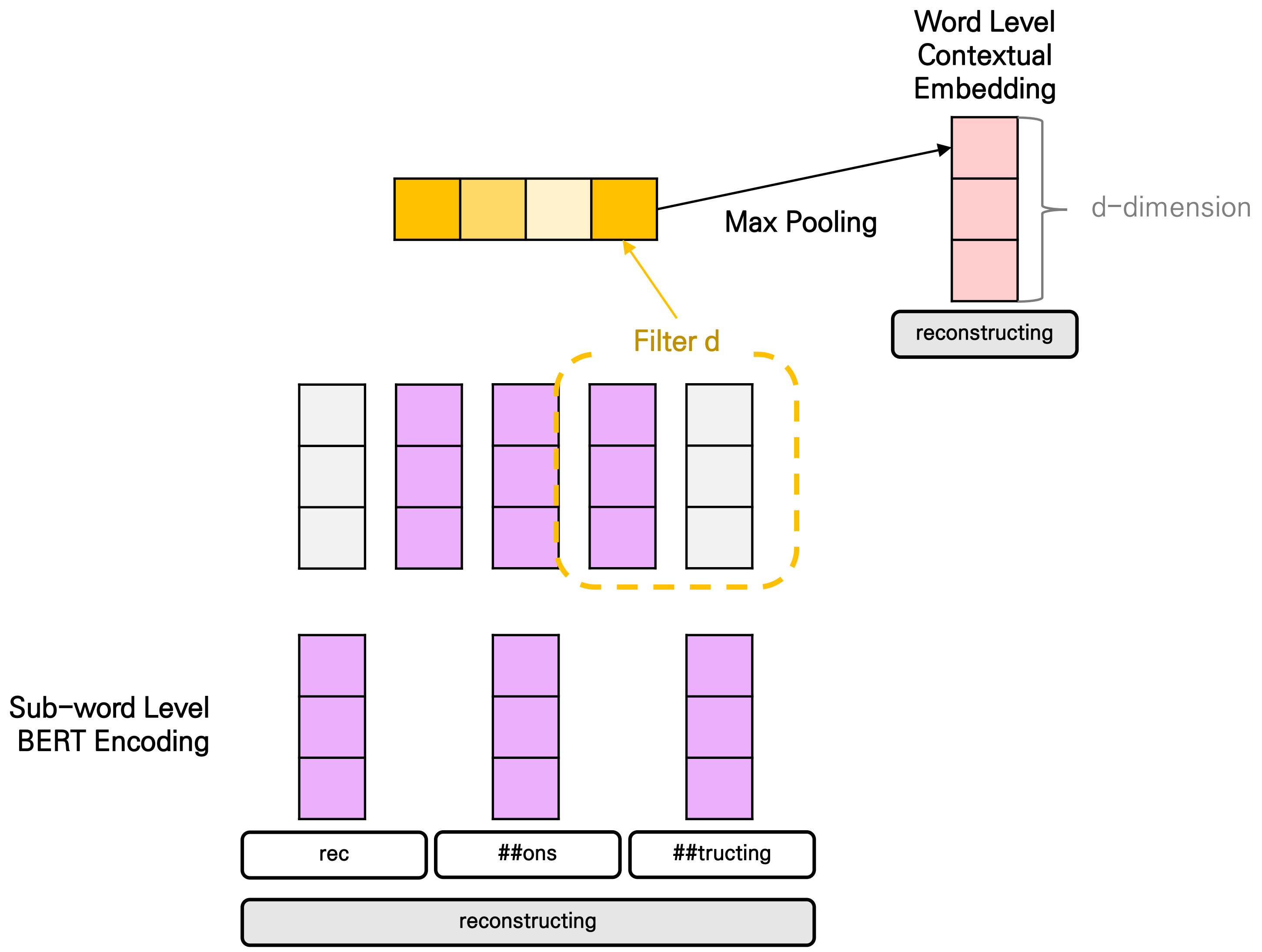
00

Appendix 1

1D Convolution

From [Paper Review] Semantics-aware BERT for Language Understanding (Myeongsup Kim, 2020)

〈1D Convolution〉

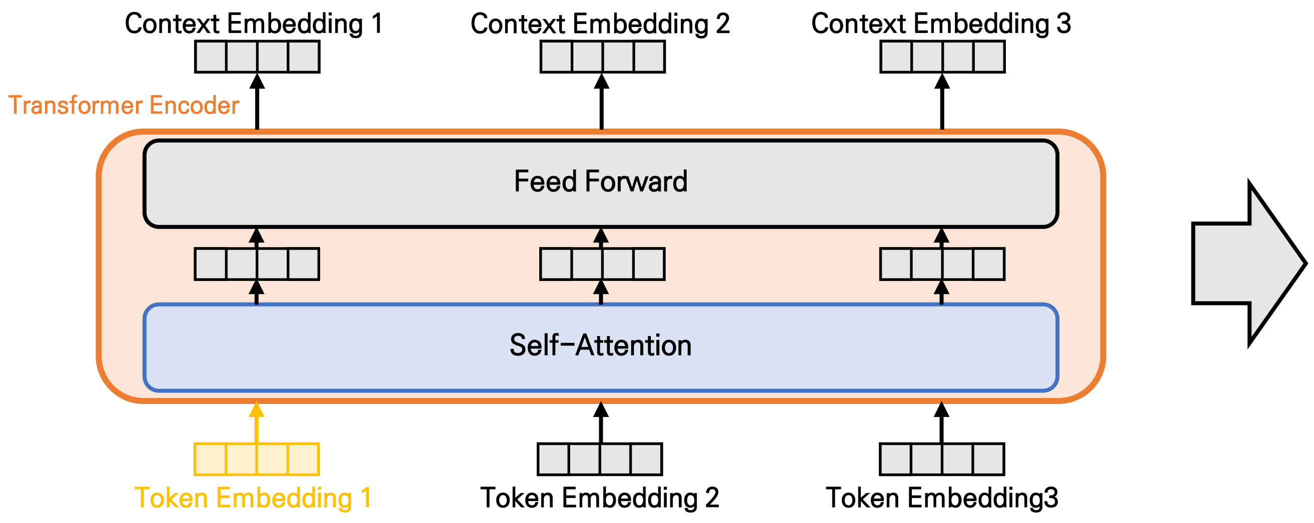


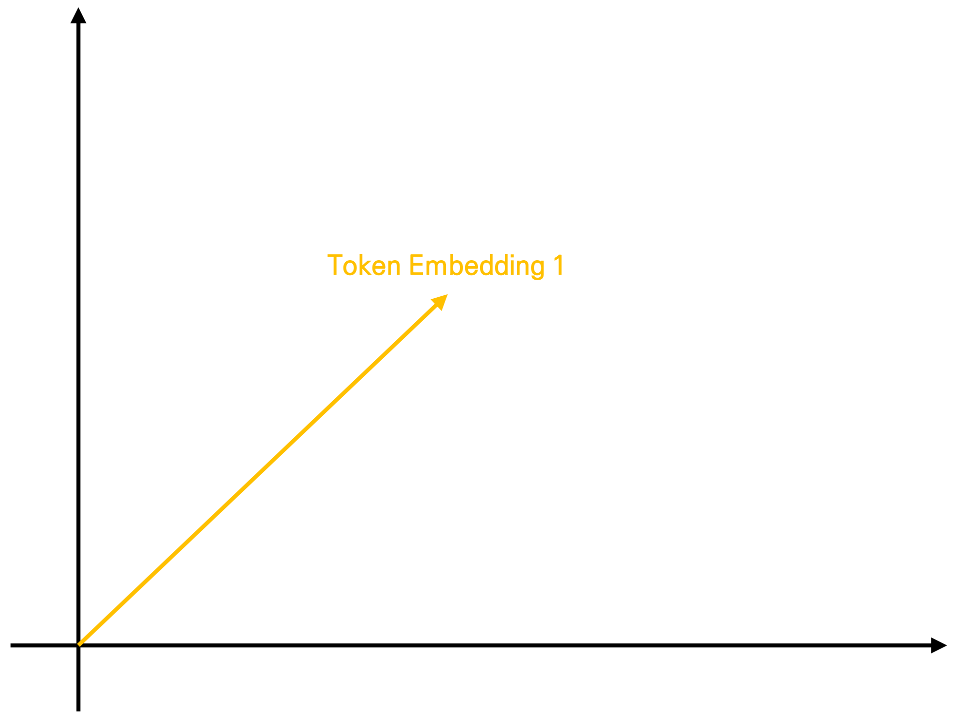
Appendix 2

Self-Attention

From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup Kim, 2021)

〈Self-Attention〉



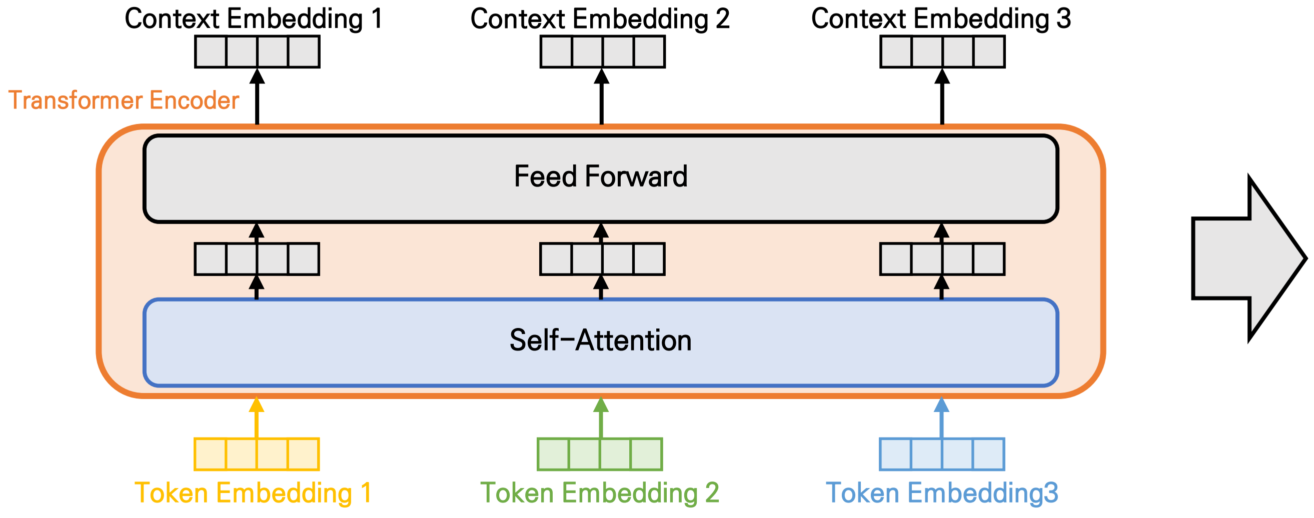


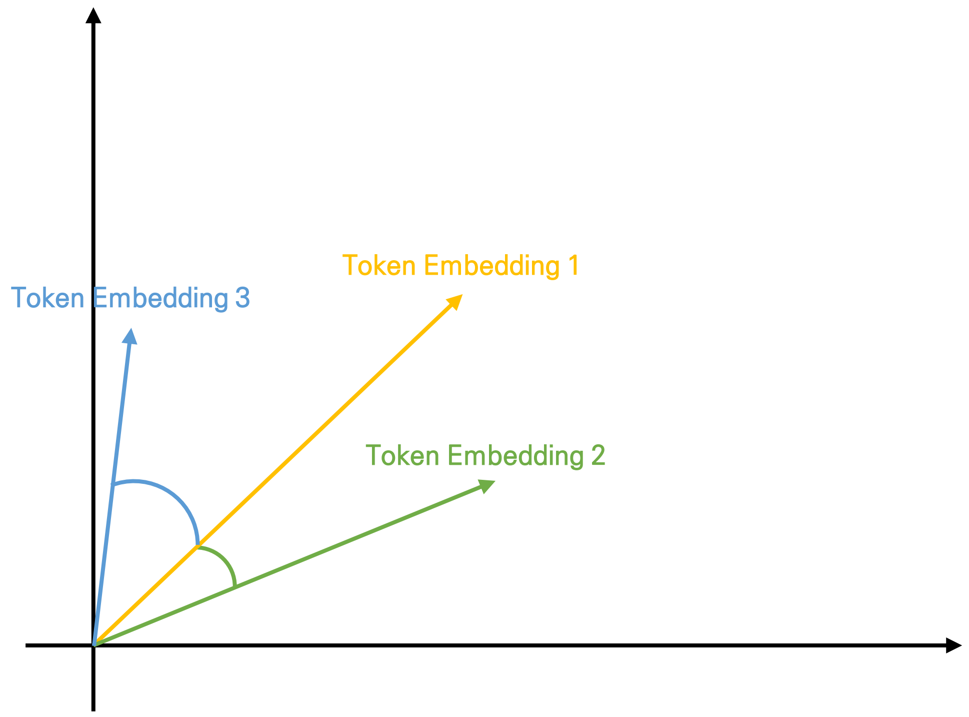
Appendix 2

Self-Attention

From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup Kim, 2021)

〈Self-Attention〉



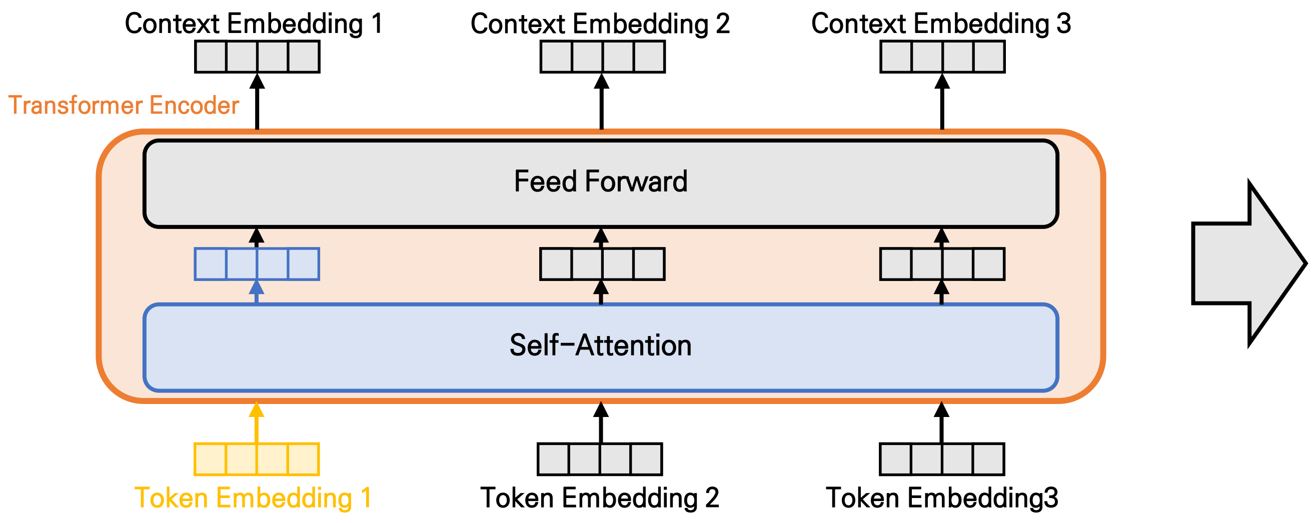


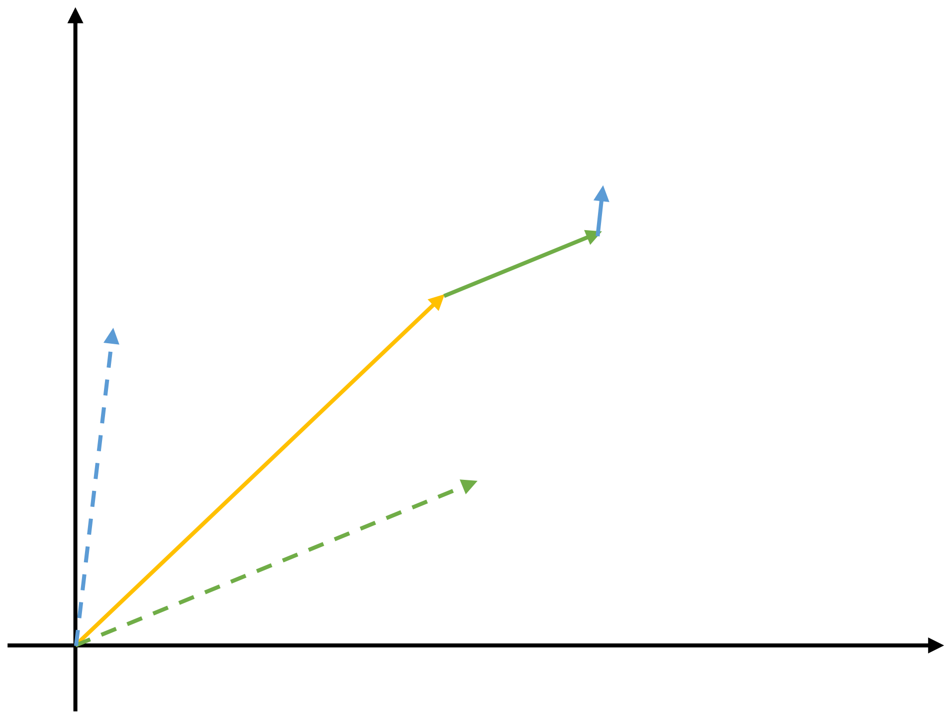
Appendix 2

Self-Attention

From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup Kim, 2021)

〈Self-Attention〉



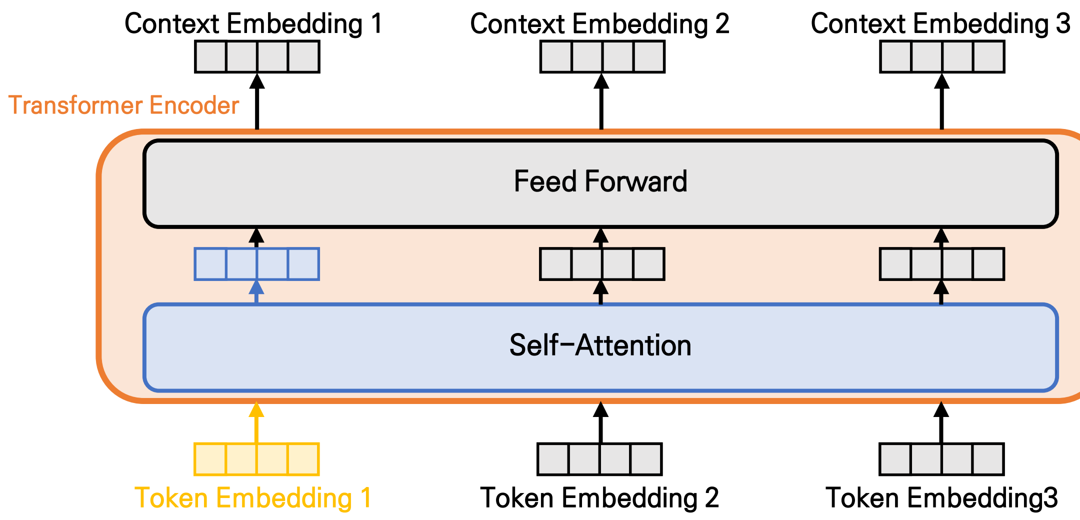


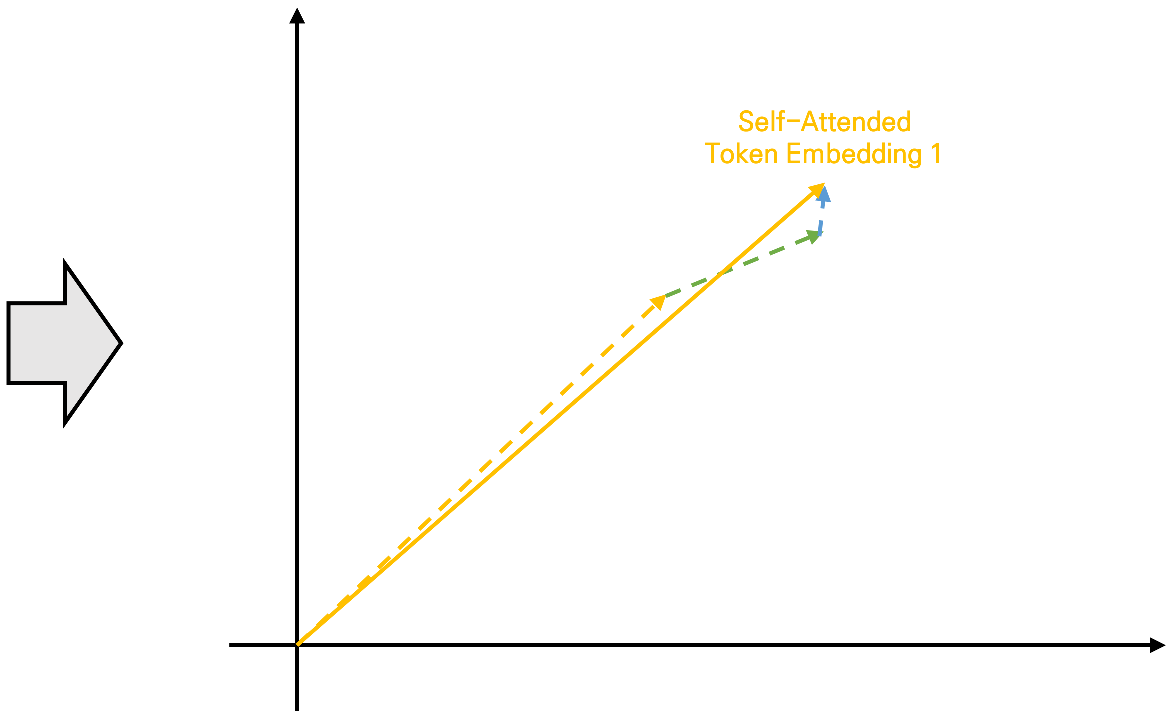
Appendix 2

Self-Attention

From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup Kim, 2021)

〈Self-Attention〉





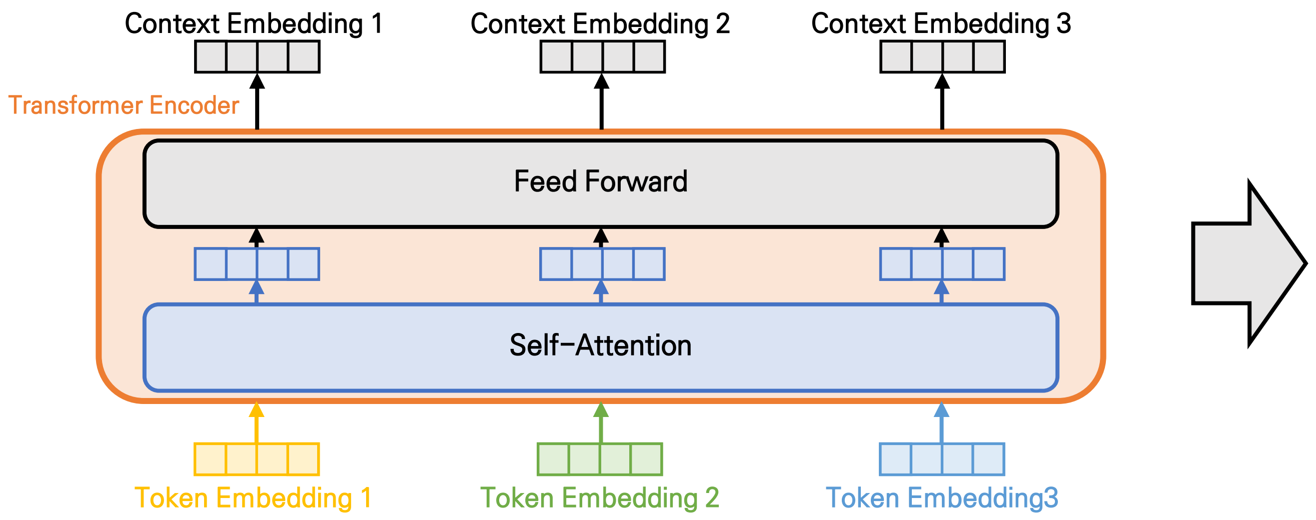
00

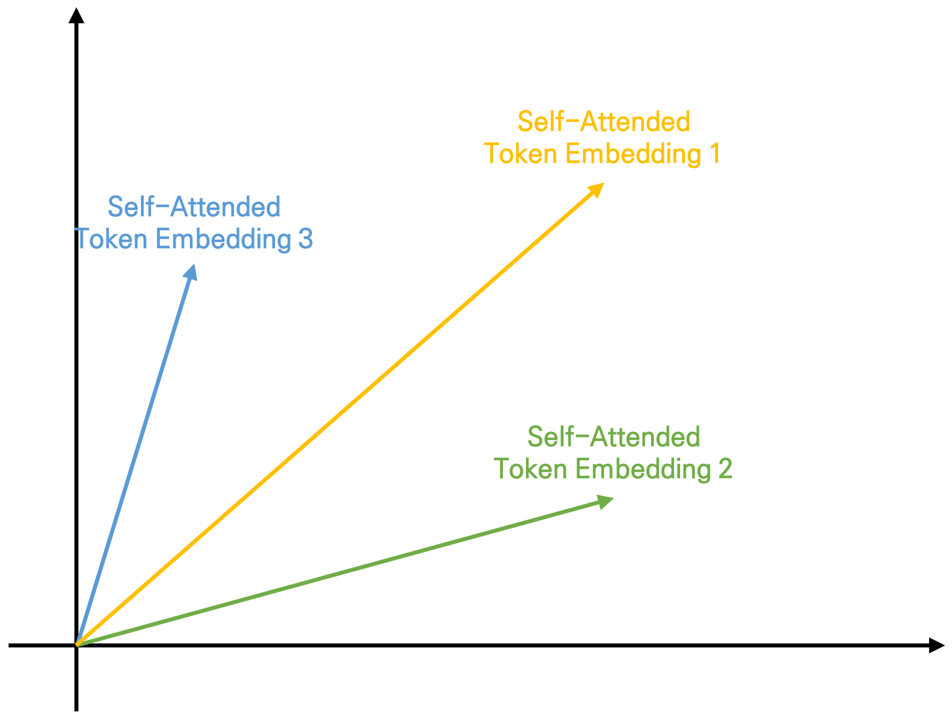
Appendix 2

Self-Attention

From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup Kim, 2021)

⟨Self-Attention⟩




Self-Attended Token Embedding 1
Self-Attended Token Embedding 3
Self-Attended Token Embedding 2

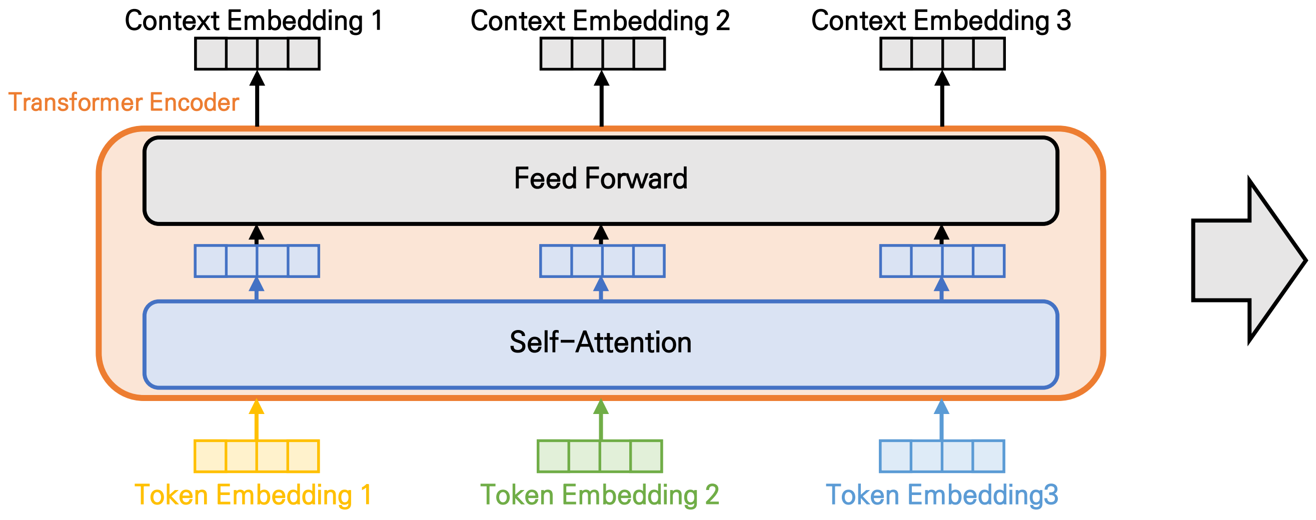
00

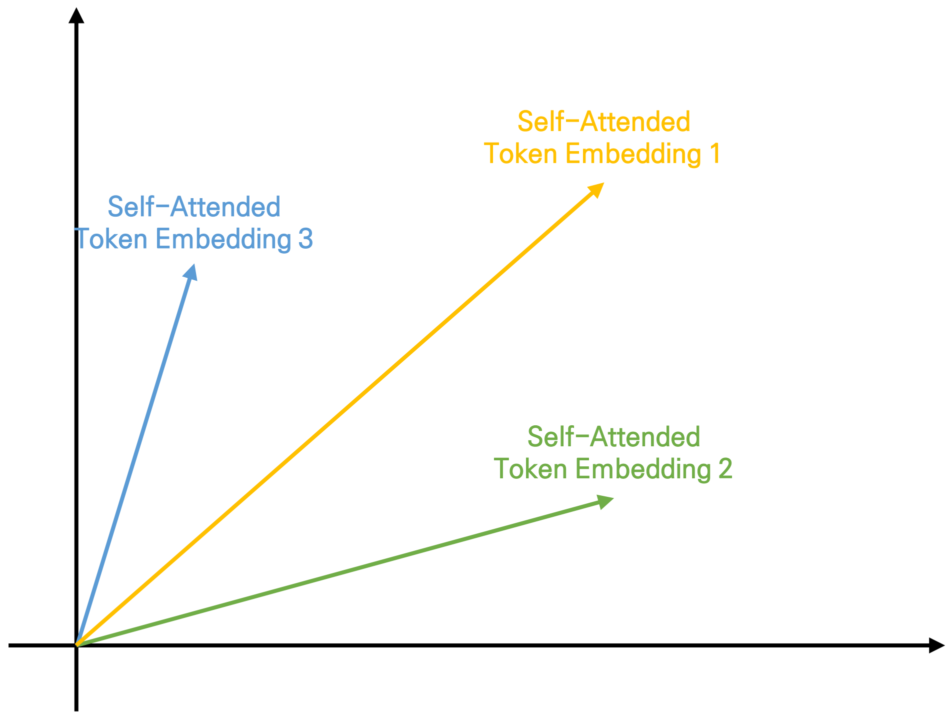
Appendix 2

Self-Attention

From [Paper Review] FreeLB: Enhanced Adversarial Training for Natural Language Understanding (Myeongsup Kim, 2021)

〈Self-Attention〉





Self-Attended Token Embedding 1
Self-Attended Token Embedding 3
Self-Attended Token Embedding 2

Appendix 3

Semantic Boundary

〈Embedding Distance and Context〉

Sentence A: "He thereupon *went* to London and spent the winter talking to men of wealth."

went: to move from one place to another **(Sense 1)**

Sentence B: "He *went* prone on his stomach, the better to pursue his examination."

went: to enter into a specified state. **(Sense 2)**

Sense 1: ***matching sense*** of Sentence A

Sense 2: ***opposing sense*** of Sentence A

00

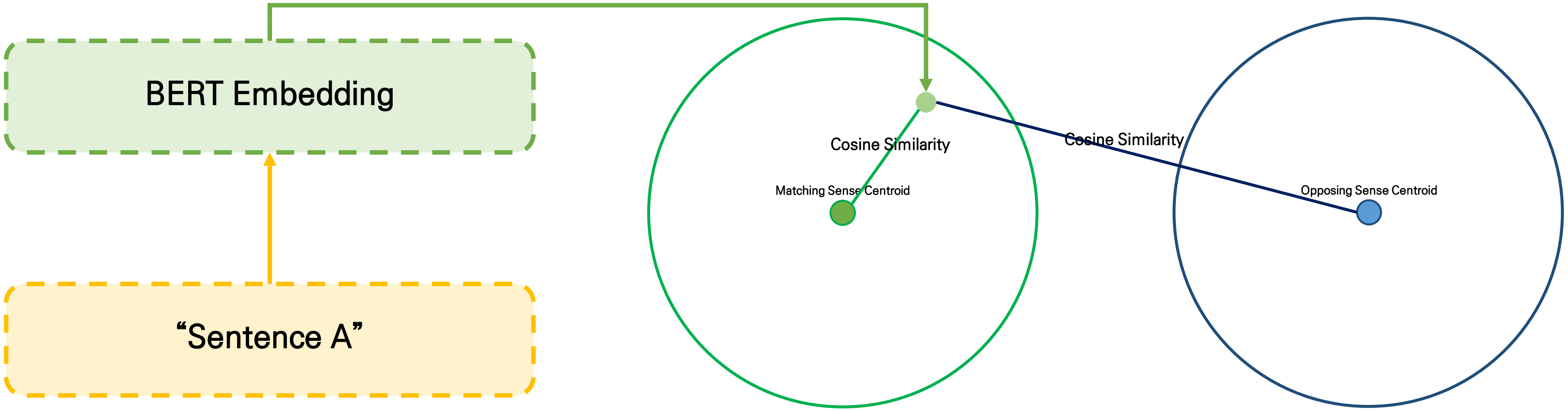
Appendix 3

Semantic Boundary

From [Paper Review] Syntax and Semantics in Language Model Representation (Myeongsup Kim, 2020)

〈Embedding Distance and Context〉

$$individual\ similarity\ ratio = \frac{matching\ sense\ similarity}{opposing\ sense\ similarity}$$



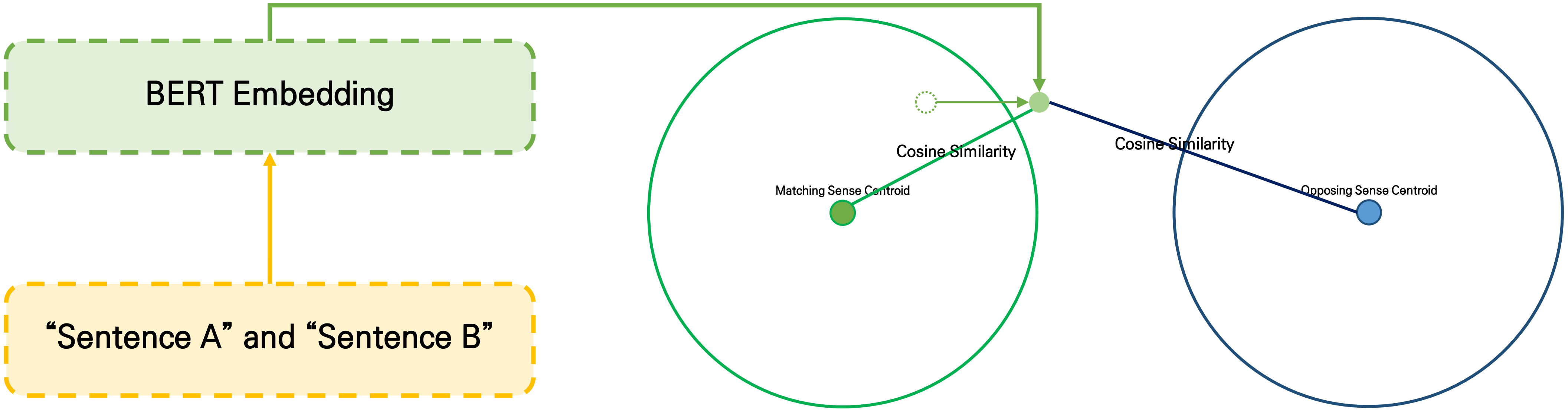
Appendix 3

Semantic Boundary

From [Paper Review] Syntax and Semantics in Language Model Representation (Myeongsup Kim, 2020)

〈Embedding Distance and Context〉

$$concatenated\ similarity\ ratio = \frac{matching\ sense\ similarity}{opposing\ sense\ similarity}$$



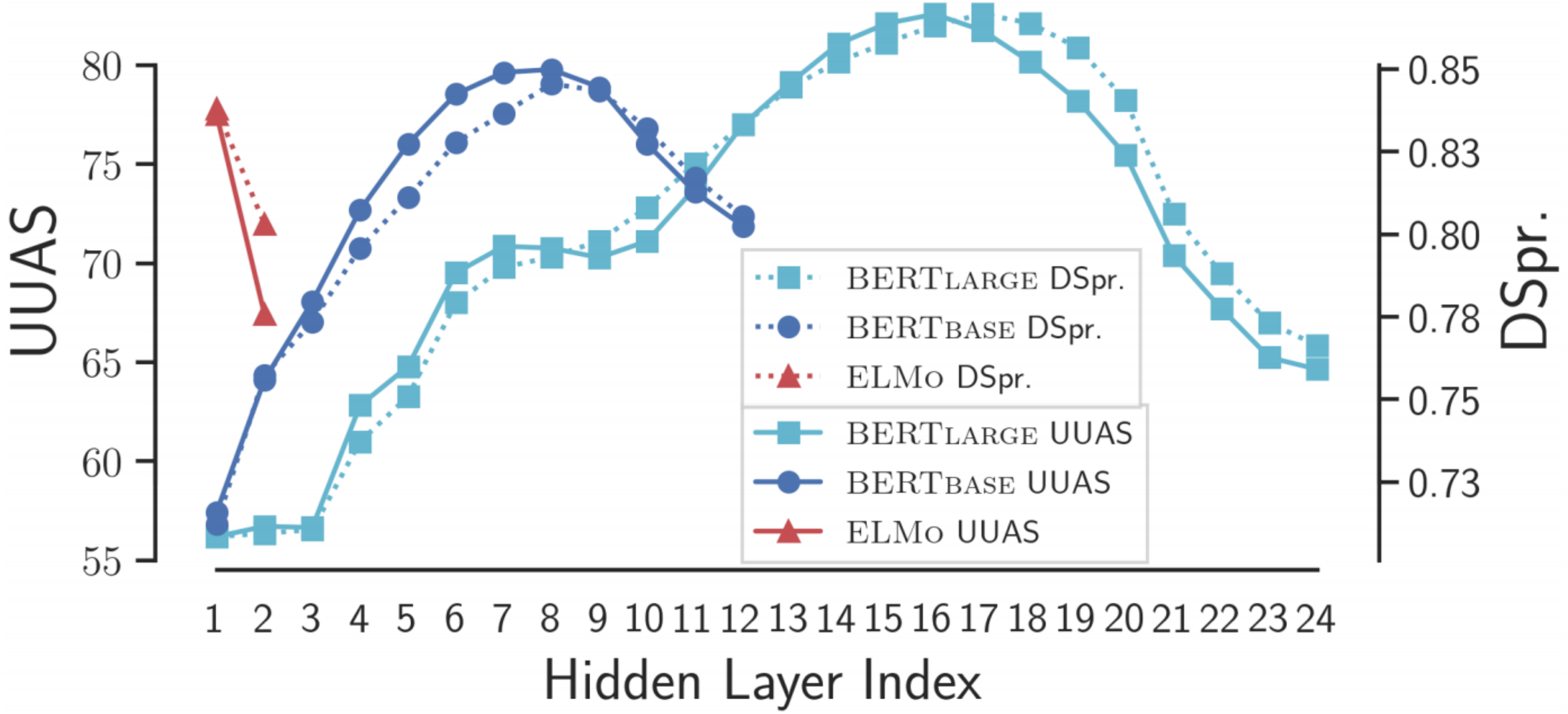
00

Appendix 4

Structural Probe

From [Paper Review] Syntax and Semantics in Language Model Representation (Myeongsup Kim, 2020)

〈Result〉



〈Parse Distance UUAS & Dspr. Across BERT and ELMo Layers〉