



# Attention Is All you need

- ◆ Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin,
- ◆ "Attention Is All You Need," NIPS 2018
- ◆ PDF 링크 <https://arxiv.org/abs/1706.03762>

정봉수

# TABLE OF CONTENTS

---

등장 배경  
연구 동기

01



핵심 기술  
주요 기술 설명

02



Model Architecture  
모델 설명

03



04

OUTPUT  
모델 실행 결과



05

FUTURE



06

QNA



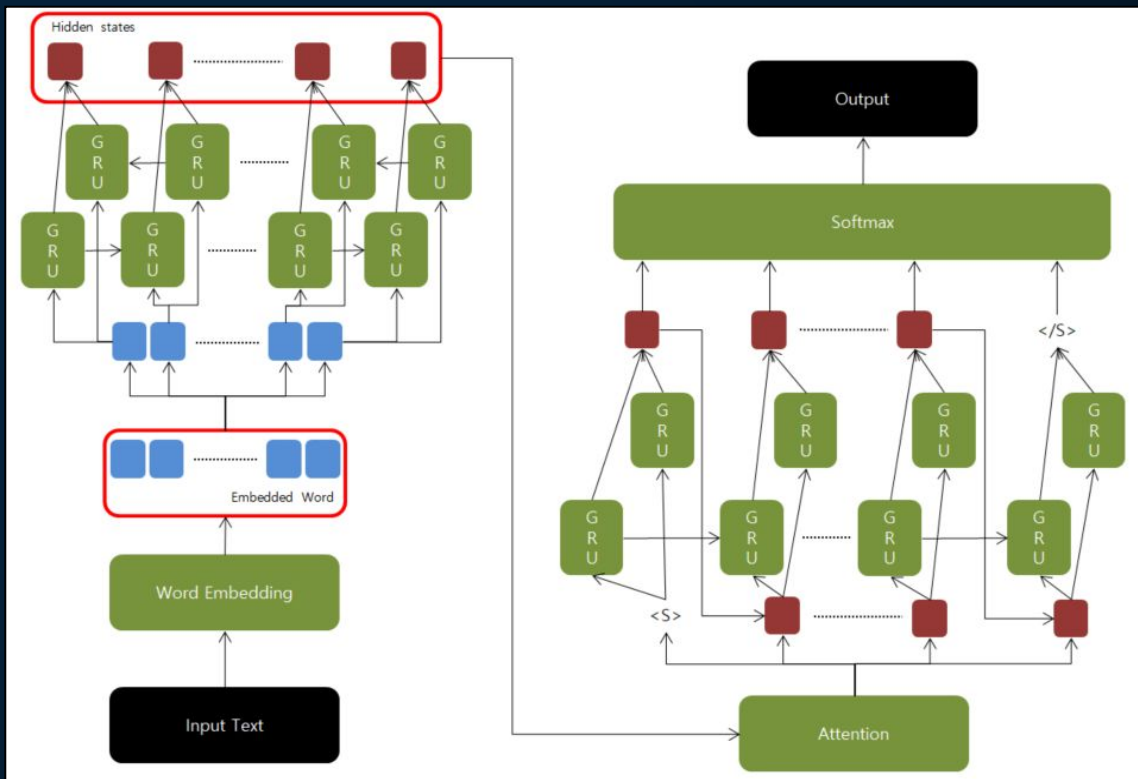
## 연구 배경

---

Sequence Data 문제(기계번역, 이미지 캡셔닝 등등) 풀때 가장 많이 사용되어지는 Encoder - Decoder 구조에 가장 많이 사용되어졌던 RNN ,  
RNN 에서 조금 발전된 RNN에 Attention 구조를 추가하는 방법론 ,  
Attention 구조를 추가함으로써 많은 발전이 있었지만 여전히 문제로 남았던  
RNN이 가지고 있는 Parallelism(병렬구조)을 막는 처리 방식,  
그것을 해결 하기 위해서 TRANSFORMER 구조가 나오게 된다.

# 기존 RNN BASE Attention 처리 방식

본 모델은 한국어 영어 번역 모델은 최근 NMT(Neural Machine Translation) 모델에 많이 활용되는 Seq2Seq 방식을 활용하여 학습하였다. Seq2Seq 방식의 NMT 모델은 입력 문장을 벡터로 변환하는 신경망(Encode)와 Encoder에서 생성된 벡터로 Attention Map 을 거쳐 번역 문장을 생성하는 신경망(Decoder)로 구성되어 있습니다.



# TRANSFORMER ARCHITECTURE

오직 Attention 과 feed forward 만으로  
구성되어져 있다.

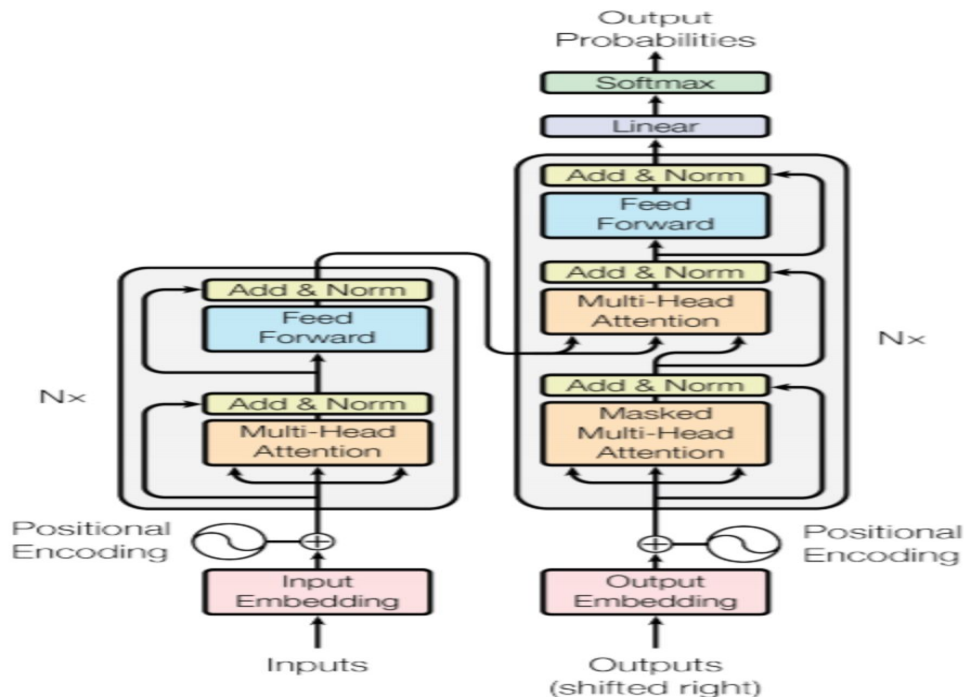


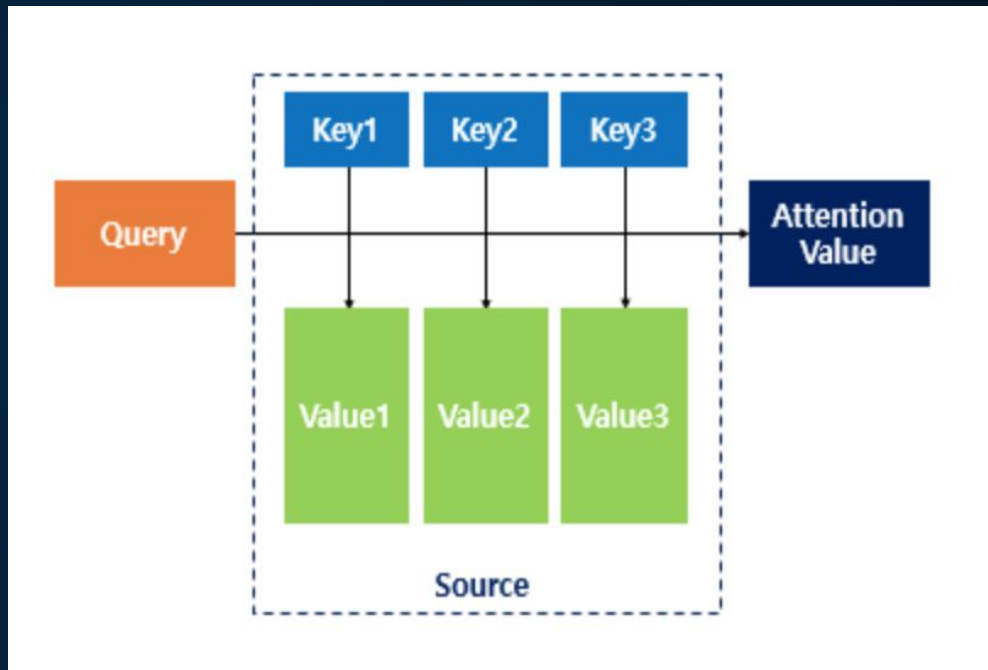
Figure 1: The Transformer - model architecture.

# Attention 이란 무엇인가 ?

Q = Query : t 시점의 디코더 셀에서의 은닉 상태

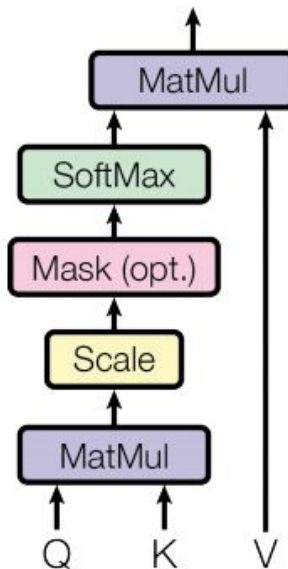
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들

V = Values : 모든 시점의 인코더 셀의 은닉 상태들



# Attention 이란 무엇인가 ?

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

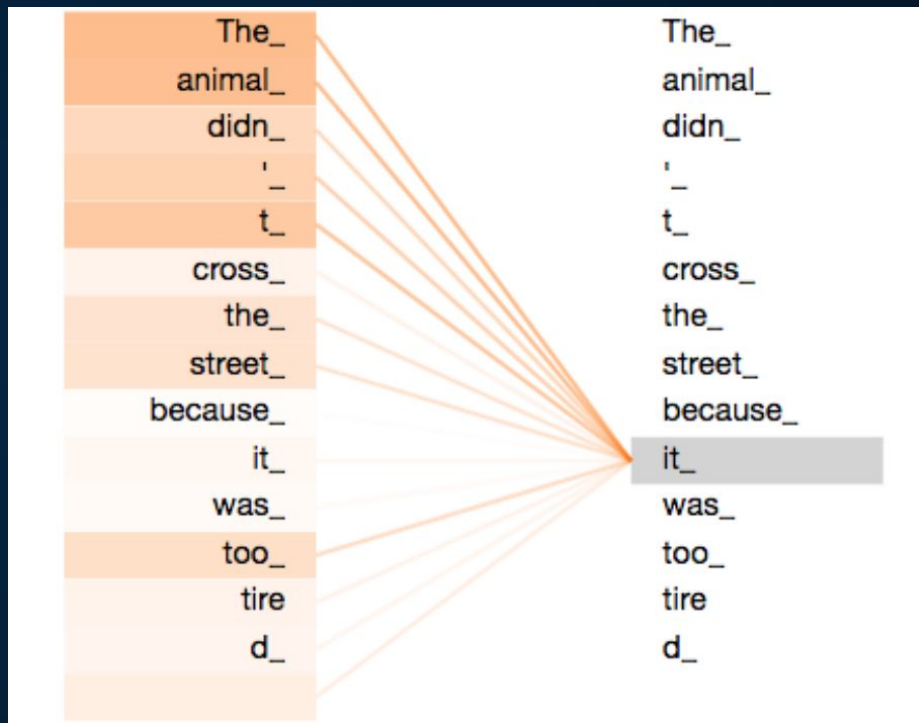


# Self Attention 이란 무엇인가 ?

앞서 설명한 Attention의 경우 Q , K , V 값이 각각

달랐다고 한다면 **self Attention**의 경우에는 이 3가지 값이 동일 하다고 생각하면 위의 개념과 완전히 일치한다.

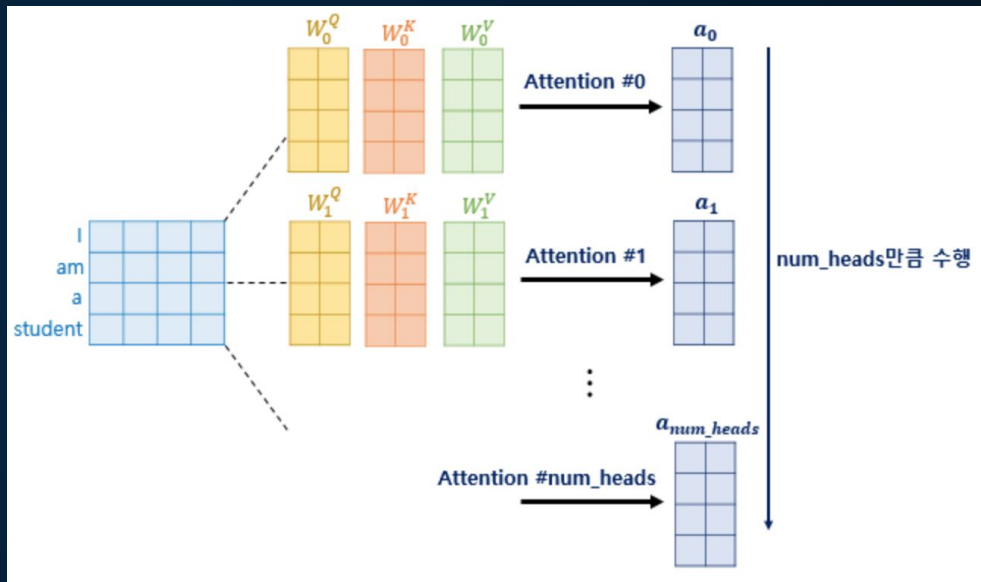
동일 하면 왜??





# Multi Head Attention

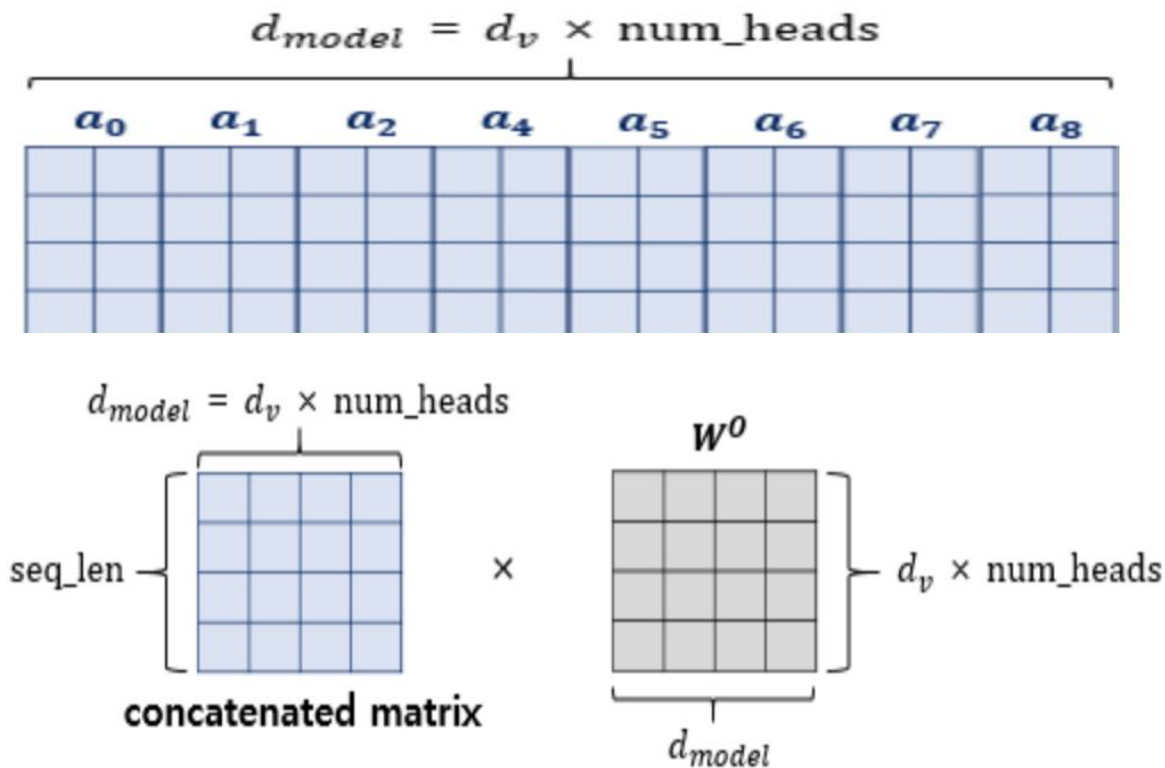
논문에서는 단일 self Attention 을 사용 하는 것보다,  
Multi Head Attention 을 사용했을 때 더 좋은 효과를 낸다고 설명한다.  
그 이유는 Multi Head 의 의미를 알면 쉽게 이해할 수 있다.  
Multi Head 란 하나의 하나의 Attention 값을 여러개의 선형 변환을 통해서 다양한 각도에서 보는 것을 의미한다.



# Multi Head Attention

Multi head를 만들기 위해서 선형변환을 시키면 차원에 변화가 일어나 기존의 Attention Value 값과는 다른 차원의 값이 나오게 된다.

이러한 문제들은 Multi head 에서 뽑아낸 Attention Value 값들을 concatenation 시킨다음 원래 Attention Value 가 가지는 차원으로 돌릴 수 있는 벡터를 다시 곱해 똑같은 크기의 행렬로 만들어서 해결한다.



# Multi Head Attention

---

그럼 **Head**의 개수는 어떻게 구하겠는가 에 대한 문제가 남아 있다.  
이 논문에서는 **Head**의 개수를 8개로 정했다.  
이 논문에서 하나의 단어를 임베딩 차원이 **512**가 되게 변환한다.  
변환된 단어가 **Multi-head-Attention**에 들어갈때  $512/\text{Head의 수} = 64$   
하나의 **head**에 임베딩 크기 **64**로 나누어진 벡터들이 들어 가게 된다.

# Positional Encoding

앞서 임베딩에 대한 이야기가 나왔다.

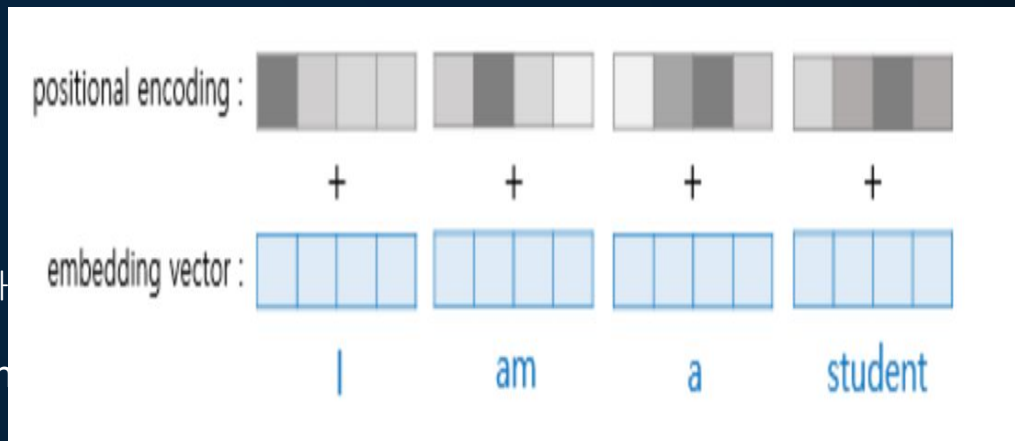
RNN의 경우 단어를 순차적으로 넣으면서  
순서를 학습한다.

하지만 Self Attention의 경우에는 순서를  
학습하는 능력이 없다.

(Attention 수식을 생각해보면 알 수 있을 것이다)

그렇기 때문에 순서를 기억하기 위해서 Position  
Encoding 이라는 기법을 사용한다.

Embedding 차원을 거쳐온 특정한 값을 더하는  
것이 전부이다.



# Positional Encoding

더하는 벡터 값은 오른쪽 수식으로 구할 수 있다.

PE 안의 변수  $pos, i$  가 있는데

$pos$ 는 입력 문장에서의 임베딩 벡터의 위치를 나타내며

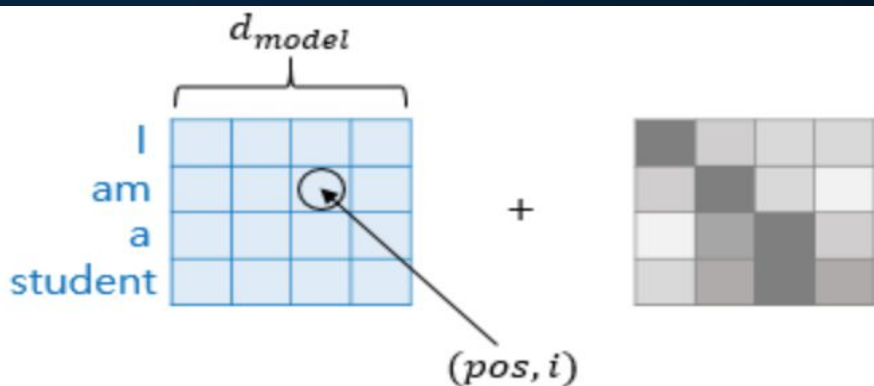
$i$ 는 임베딩 벡터 내의 차원의 인덱스를 의미한다.

예시로 아래의 그림에서  $pos = 2, i = 3$  이된다.

수식을 잘 보게 되면 짝수 인덱스를 가진것은  $\sin$  홀수 인덱스를 가진것은  $\cos$  값을 더해주고 있다.

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$



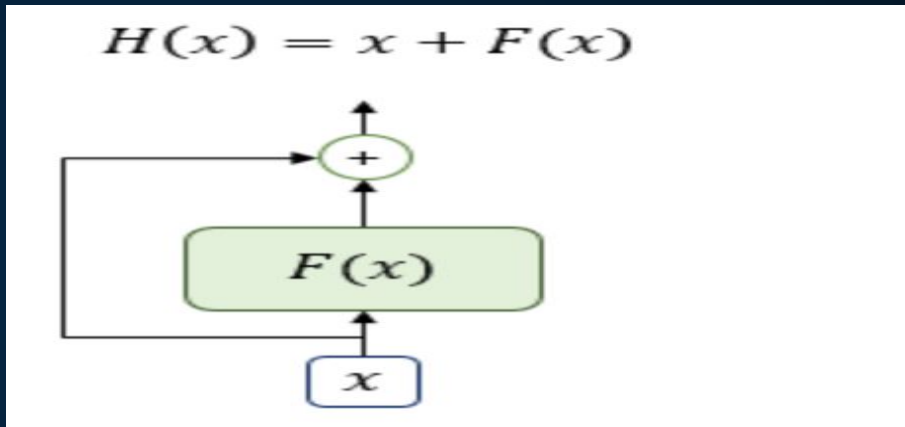
# Residual Connection

앞선 Multi-head Attention의 설명에서 차원을 맞춰 준다는 설명을 한적이 있다.

이는 이 Layer에서만 해당되는 것이 아니고 Transformer의 모든 Layer을 거치면 동일한 차원으로 결과가 나오게 설계를 하였다.

이는 Residual connection과 Layer Normalization 때문이라고 논문은 설명하고 있습니다.

위의 그림이 Residual connection의 함수 그래프입니다. 그림 그대로 input과 어떤 함수에 input을 넣어서 나온값을 더해 준것을 잔차 연결이라고합니다.




$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

# OUTPUT

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.8</b>	$2.3 \cdot 10^{19}$	

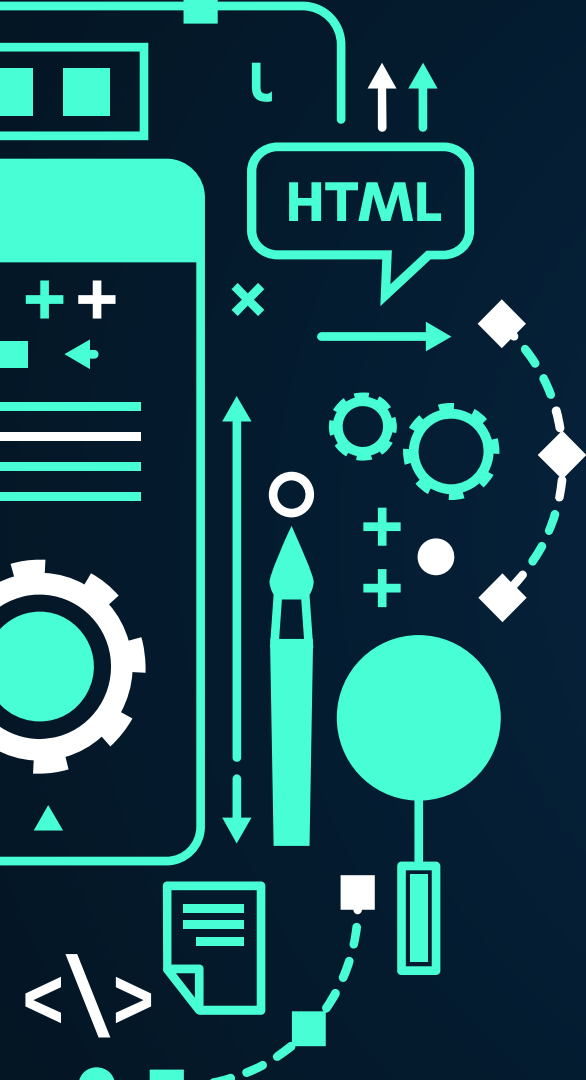


예전에 모 대학의 수학과 교수님에게 “어떤 수학 전공자를 인공지능 업계에서 선호합니까?”라고 물어 본 적이 있습니다. 그 분께서는 ‘어떤 전공’이 중요한지에 대한 답을 주시지 않고, “특정 전공보다 소통 능력이 더 중요합니다”라고 말한 바 있습니다. 이렇듯 어떠한 분야에서도 의사소통이 중요시 되는 만큼 모델이 발전해 더 좋은 결과를 얻을 수 있었으면 좋겠습니다.

**—SOMEONE FAMOUS**







# THANKS!

Does anyone have any question?

hmy831004@google.com  
010 2687 6629



# RESOURCES

- Effective approaches to attention-based neural machine translation  
(Minh-Thang Luong, Hieu Pham, Christopher D. Manning)
- D. Bahdanau, K. Cho, and  
Y. Bengio. 2015. Neural machine translation by  
jointly learning to align and translate. In ICLR.  
(Bahdanau et al.2015 )
- BLEU: a method for automatic evaluation of machine translation  
(Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. J. (2002))
- Re-evaluating the Role of BLEU in Machine Translation Research  
(Callison-Burch, C., Osborne, M. and Koehn, P. (2006))
- Attention Mechanism  
(<https://wikidocs.net/22893>)