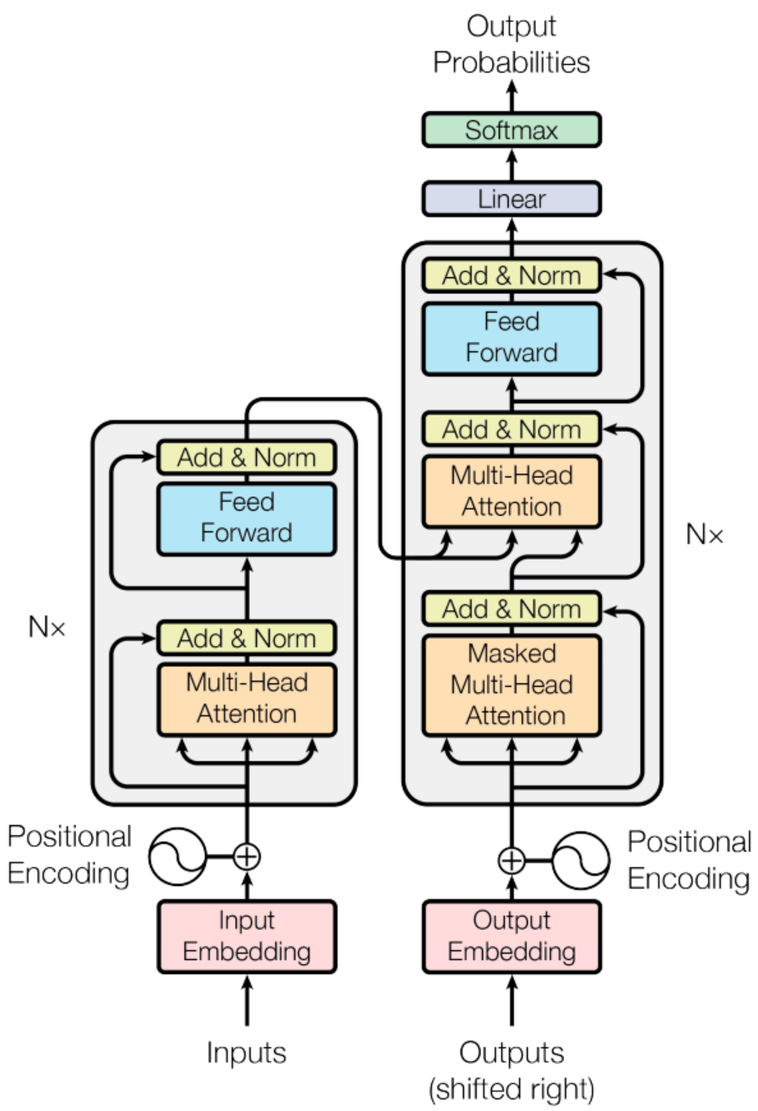
트랜스포머 최종 정리

CORE.TODAY

이제 눈에 익숙한 우리 트랜스포머



6 Dec 2017

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

https://arxiv.org/pdf/1706.03762.pdf

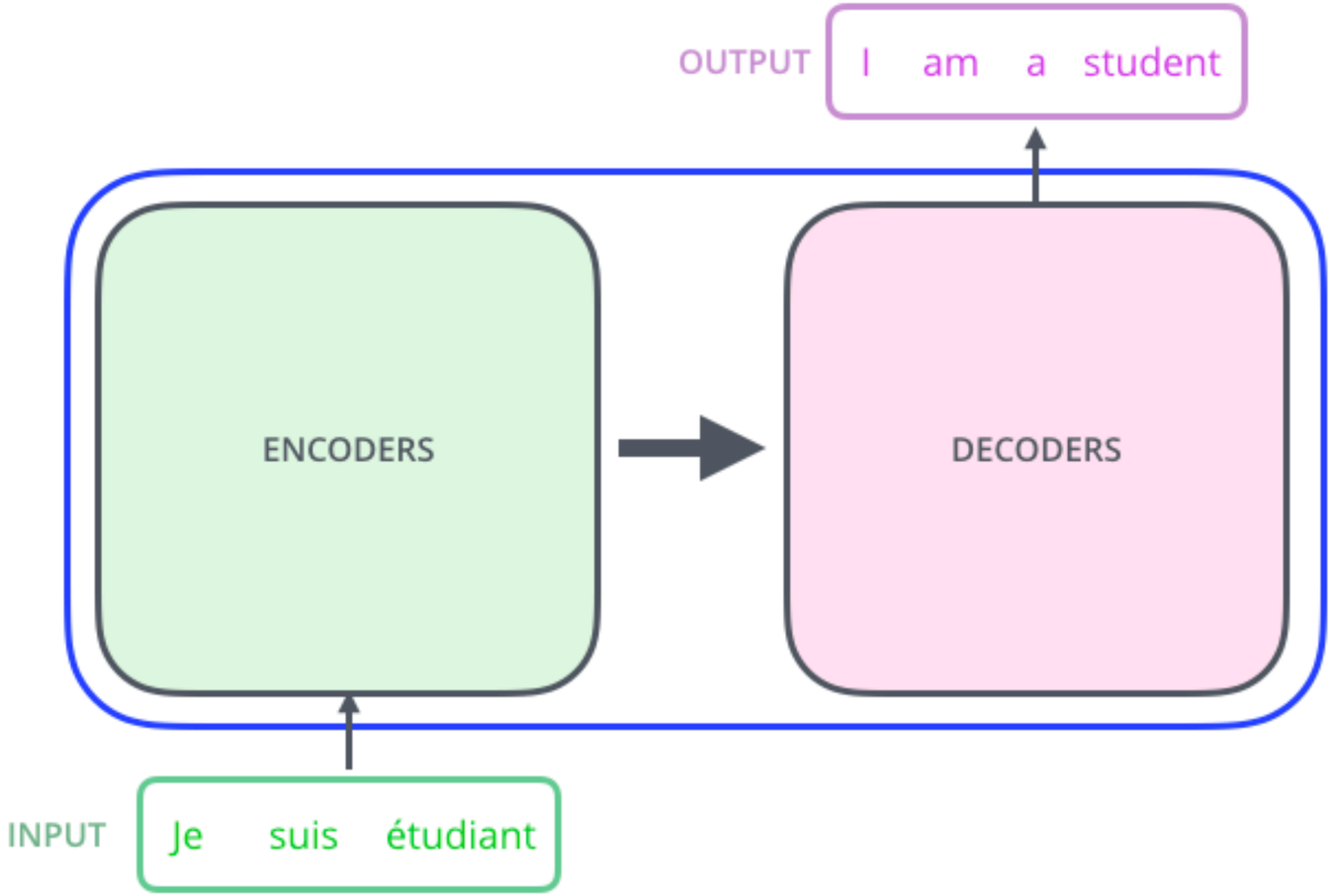
https://core.today

Transformer - 기계 번역의 경우

https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

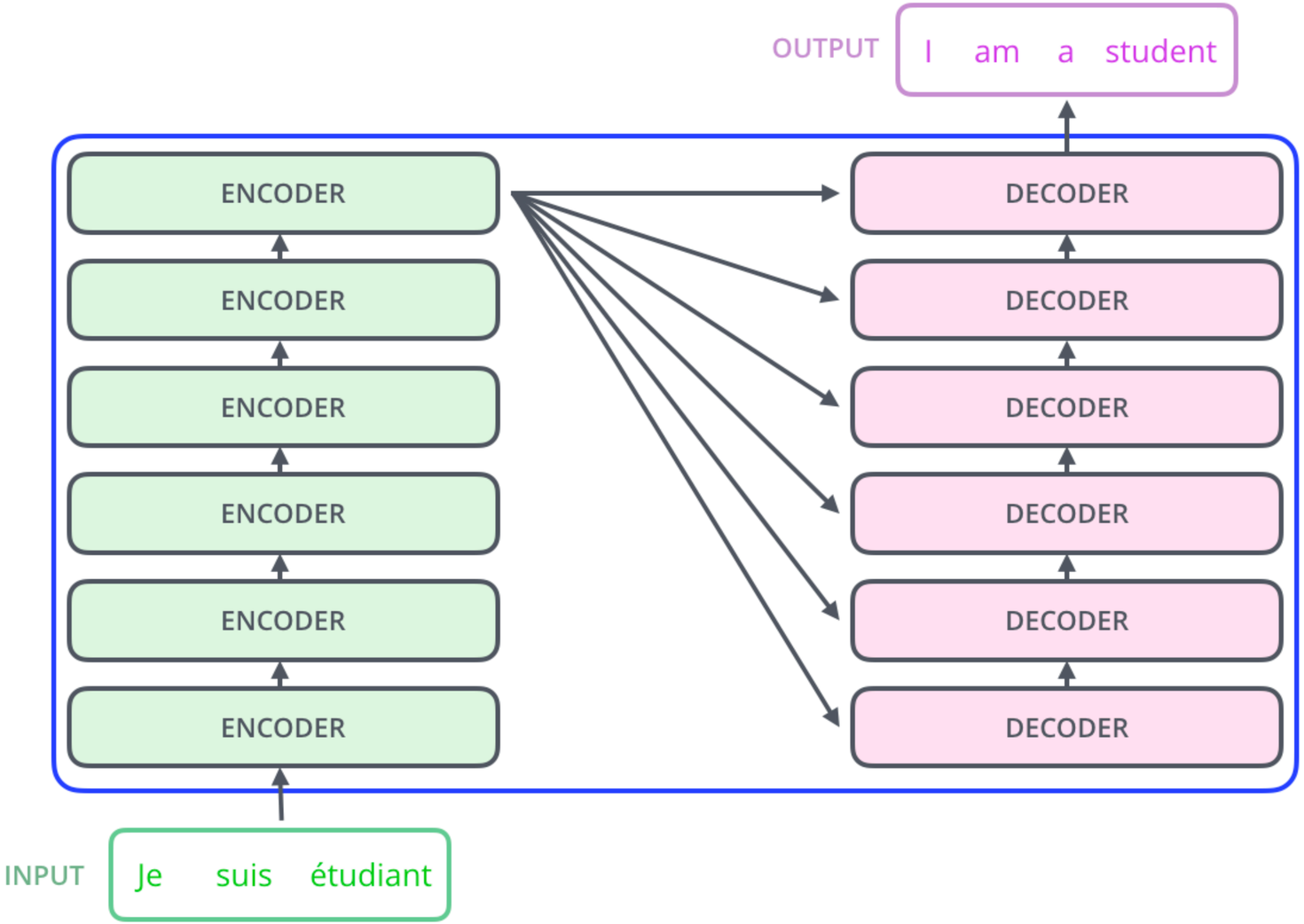
Transformer - 기계 번역의 경우



https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

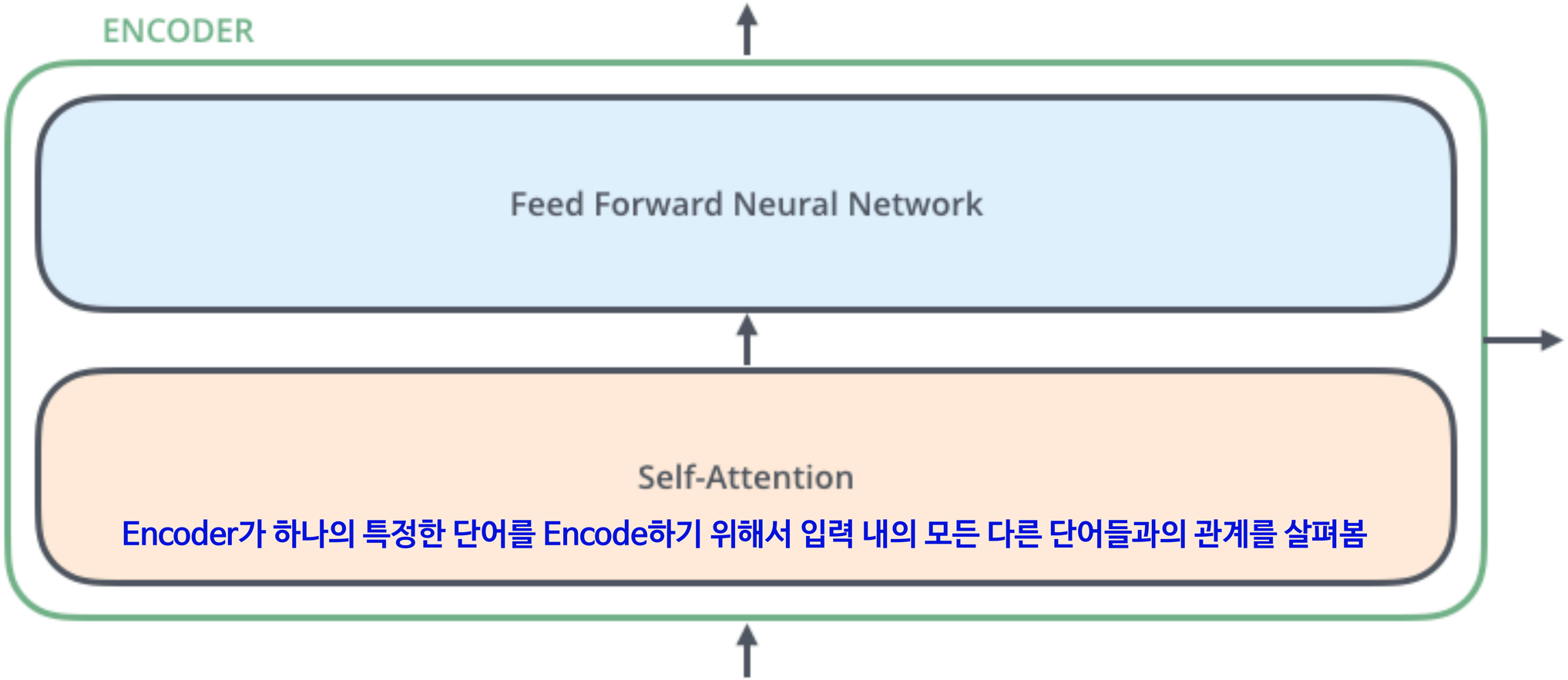
Transformer - 기계 번역의 경우



https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

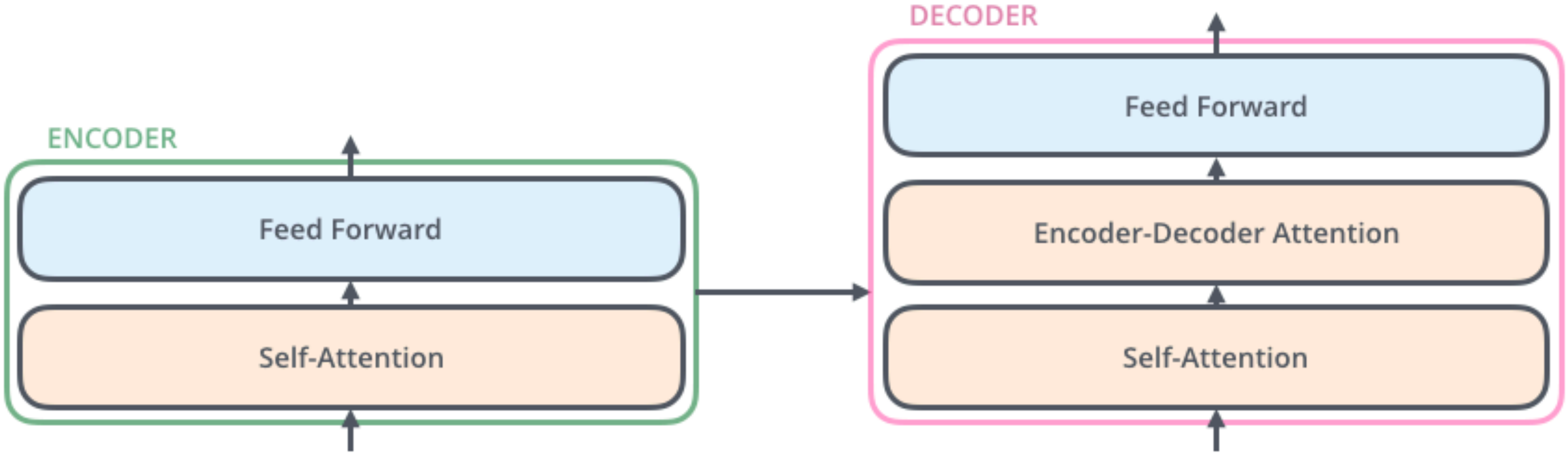
Transformer - 기계 번역의 경우



https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

Transformer - 기계 번역의 경우



Encoder-Decoder Attention :
Decoder가 입력 문장 중에서 각 타임 스텝에서
가장 관련 있는 부분에 집중할 수 있도록 함

https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

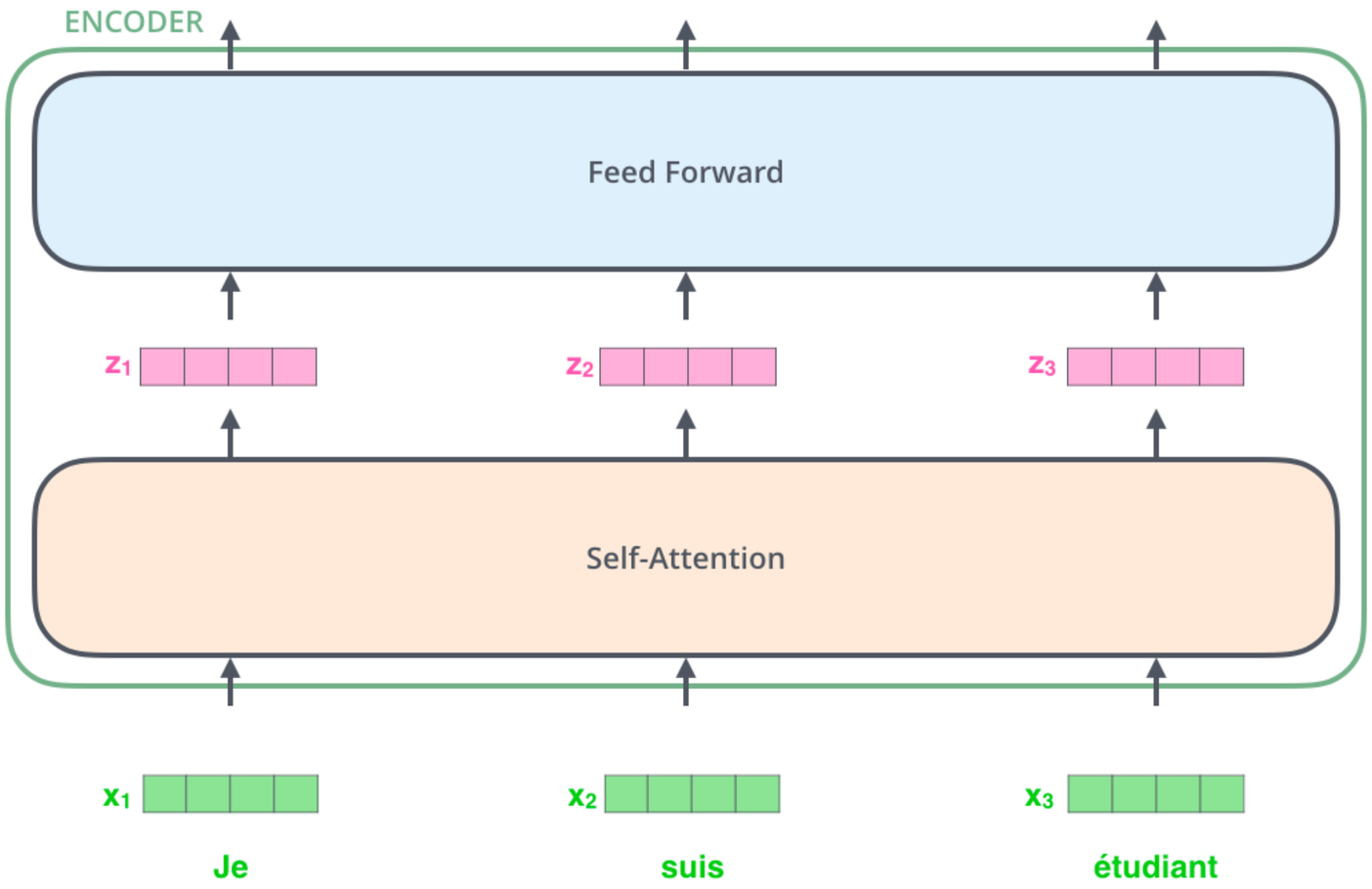
트랜스포머의 입력을 다시 살펴 보겠습니다

각 단어들을 크기 512의 벡터로 임베딩

https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

트랜스포머의 인코더를 거치며 self-encoding 과정을 거칩니다

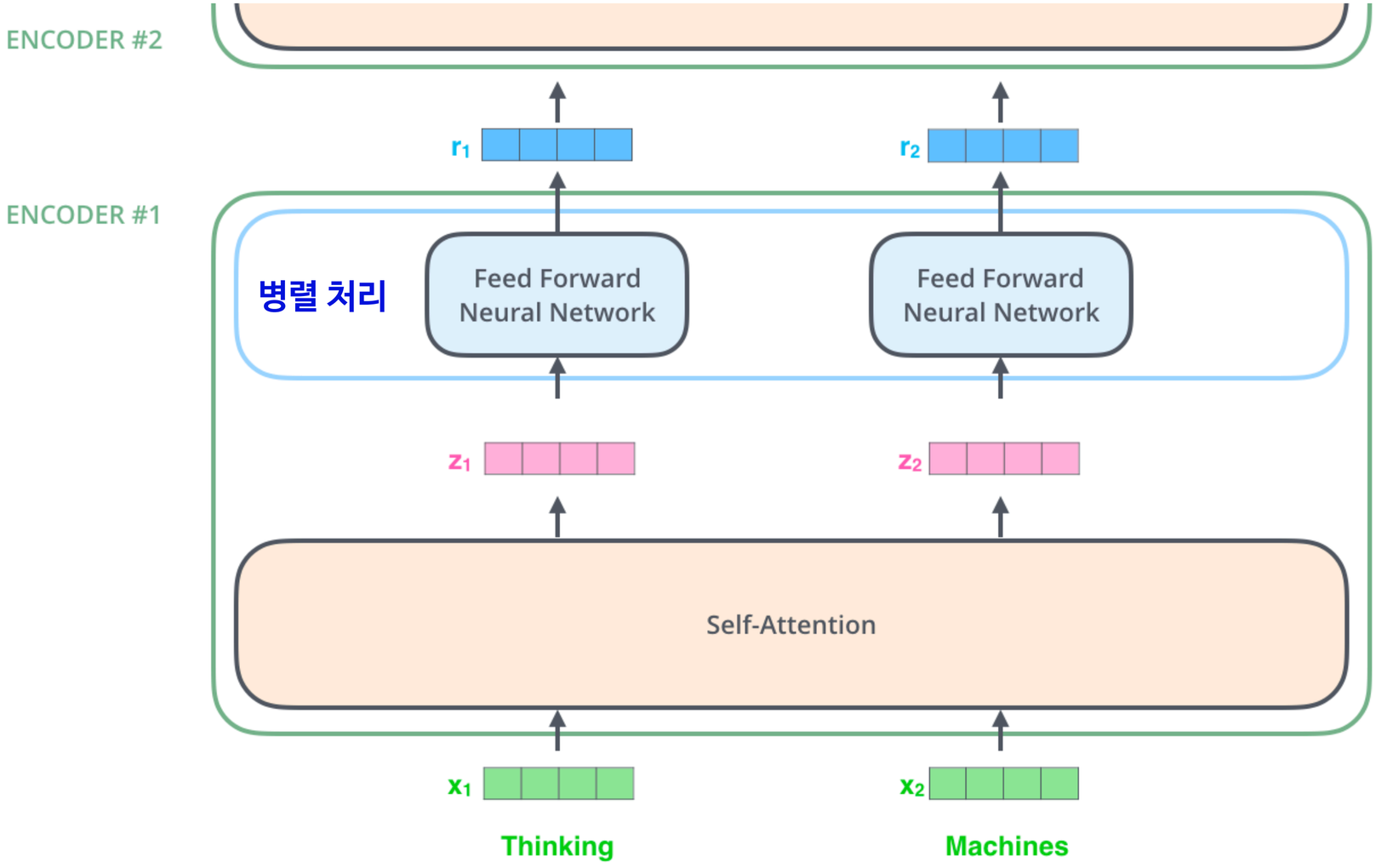


https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

Transformer - 기계 번역의 경우

FFNN은 의존성이 없어 병렬 처리가 가능합니다

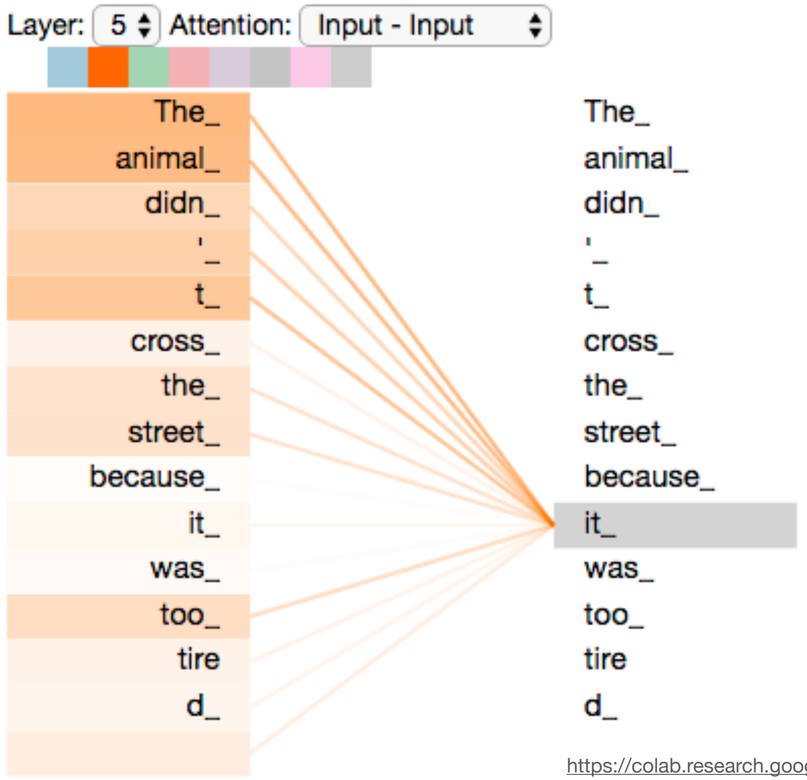


https://jalammar.github.io/illustrated-transformer/

https://nlpinkorean.github.io/illustrated-transformer/

Self-Attention은 무엇이냐구요?

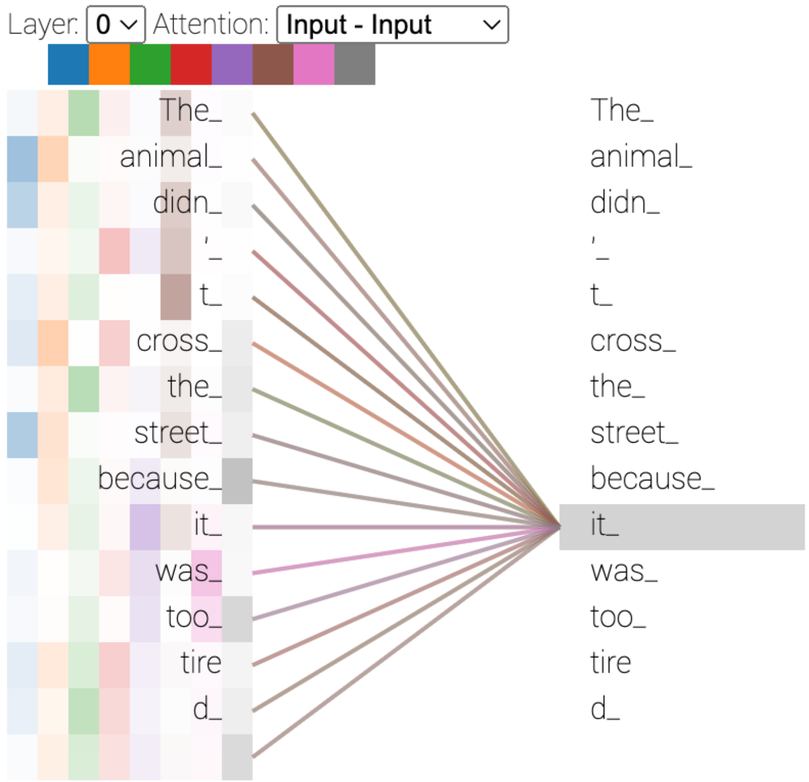
The **animal** didn't cross the street because **it** was too tired



https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

Self-Attention은 무엇이냐구요?

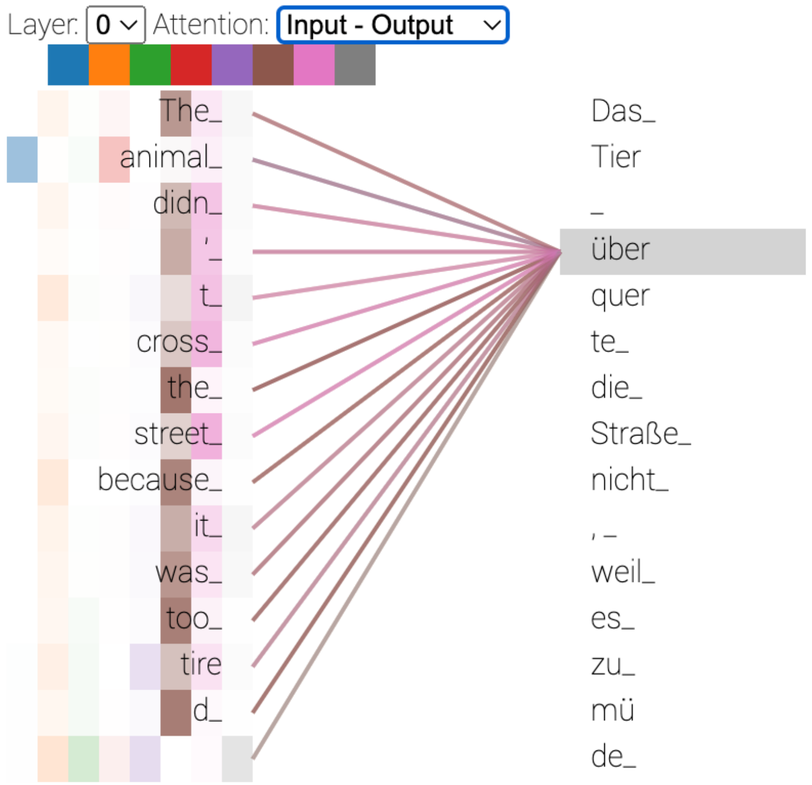
The **animal** didn't cross the street because **it** was too tired



https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

Self-Attention은 무엇이냐구요?

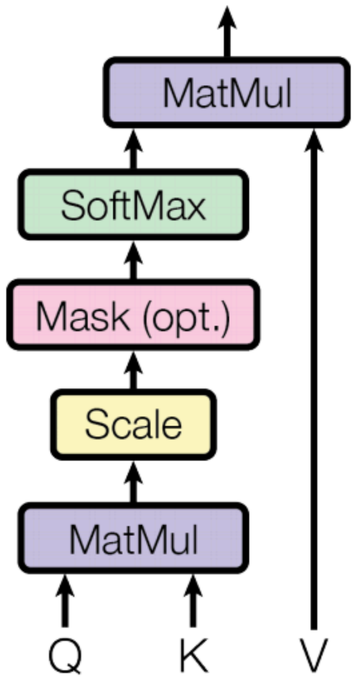
The **animal** didn't cross the street because **it** was too tired



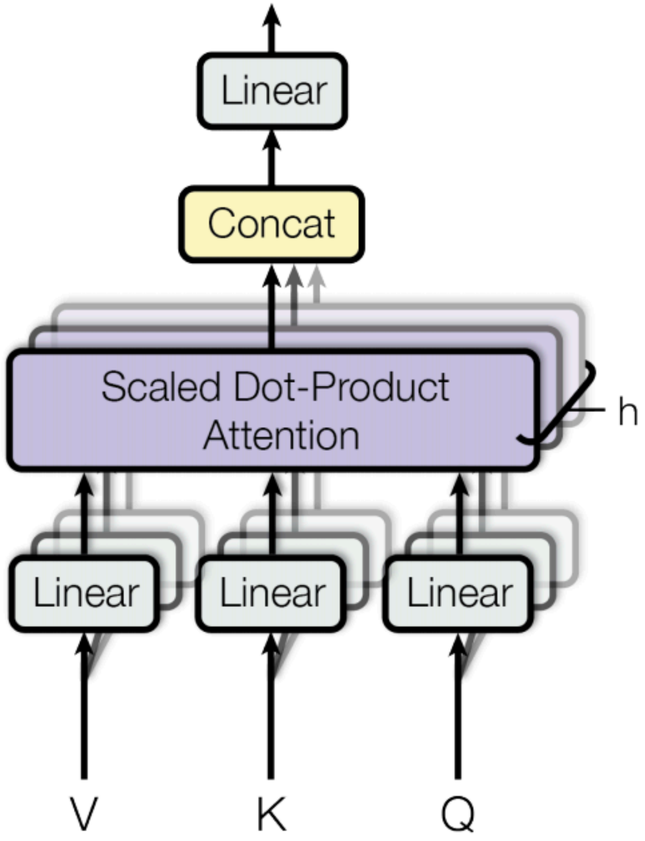
https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

멀티-헤드 어텐션

Scaled Dot-Product Attention

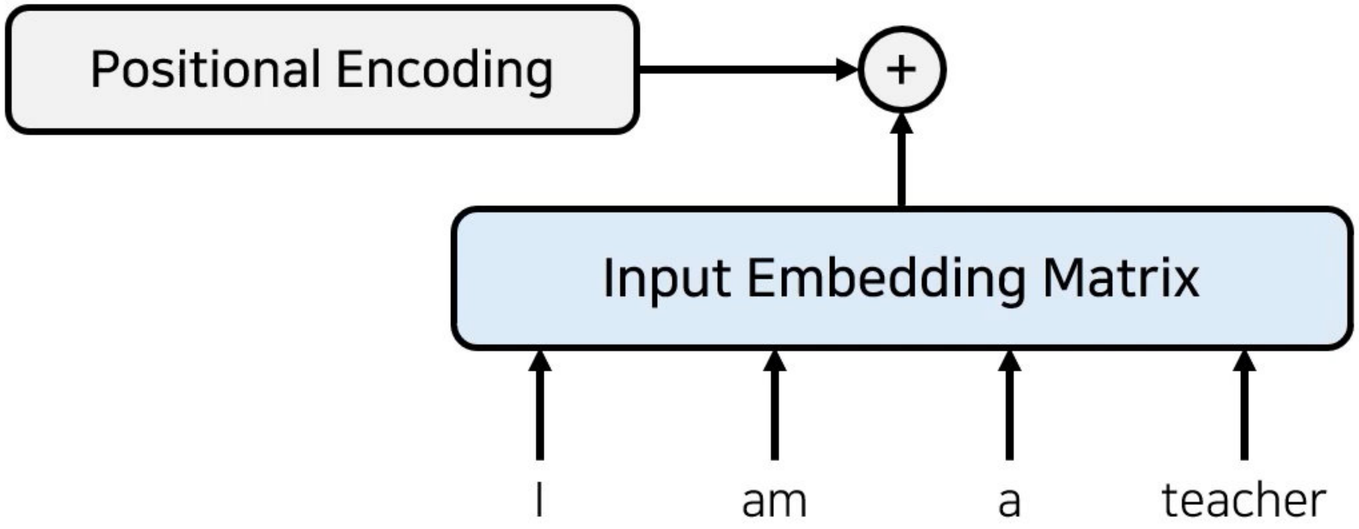


Multi-Head Attention



위치 인코딩으로 단어의 순서 정보를 반영합니다

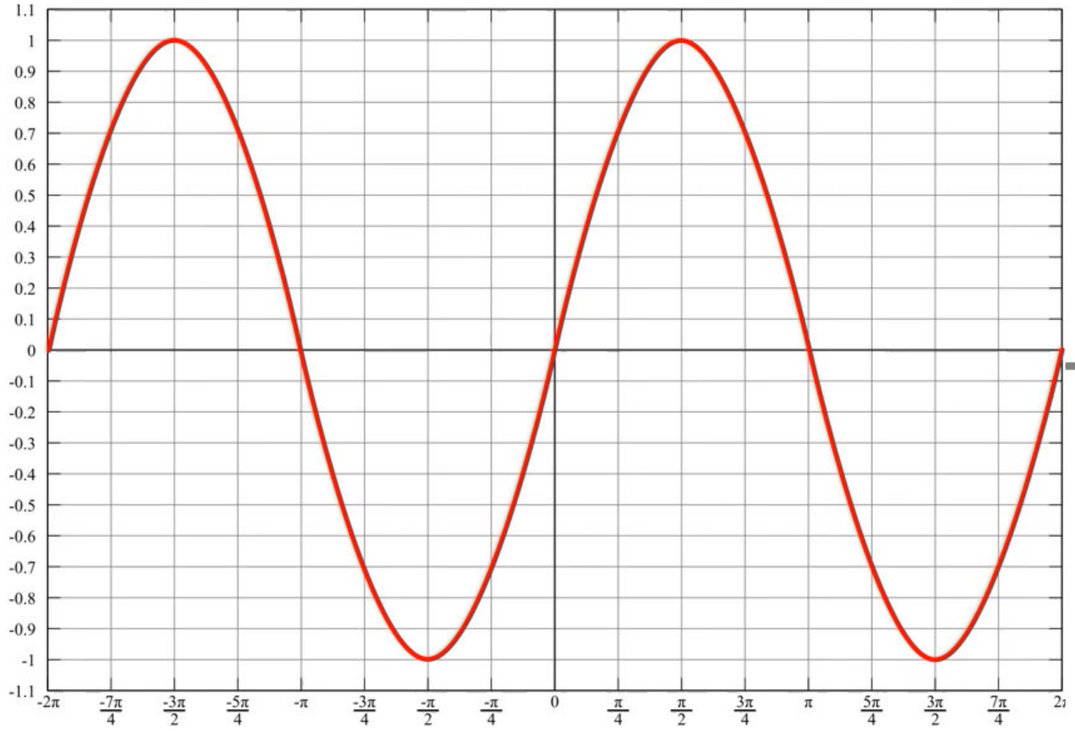
- RNN을 사용하지 않으려면 위치 정보를 포함하고 있는 임베딩을 사용해야 합니다.
  - 이를 위해 트랜스포머에서는 Positional Encoding을 사용합니다.



https://www.youtube.com/watch?v=4DzKM0vgG1Y

위치 정보는 sin과 cos으로 인코딩

- Positional Encoding은 다음과 같이 주기 함수를 활용한 공식을 사용합니다.
  - 각 단어의 **상대적인 위치 정보를 네트워크에게 입력**합니다.



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

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

: 사인(sine) 주기 함수 예시

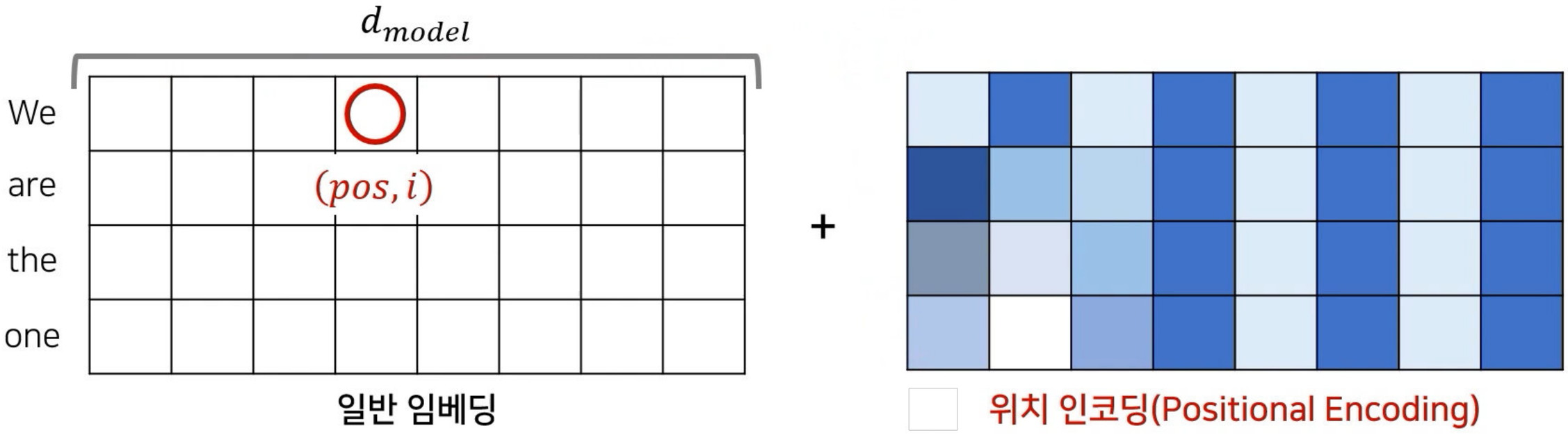
https://www.youtube.com/watch?v=4DzKM0vgG1Y

Transformer - 기계 번역의 경우

- Positional Encoding은 다음과 같이 주기 함수를 활용한 공식을 사용합니다.

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

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

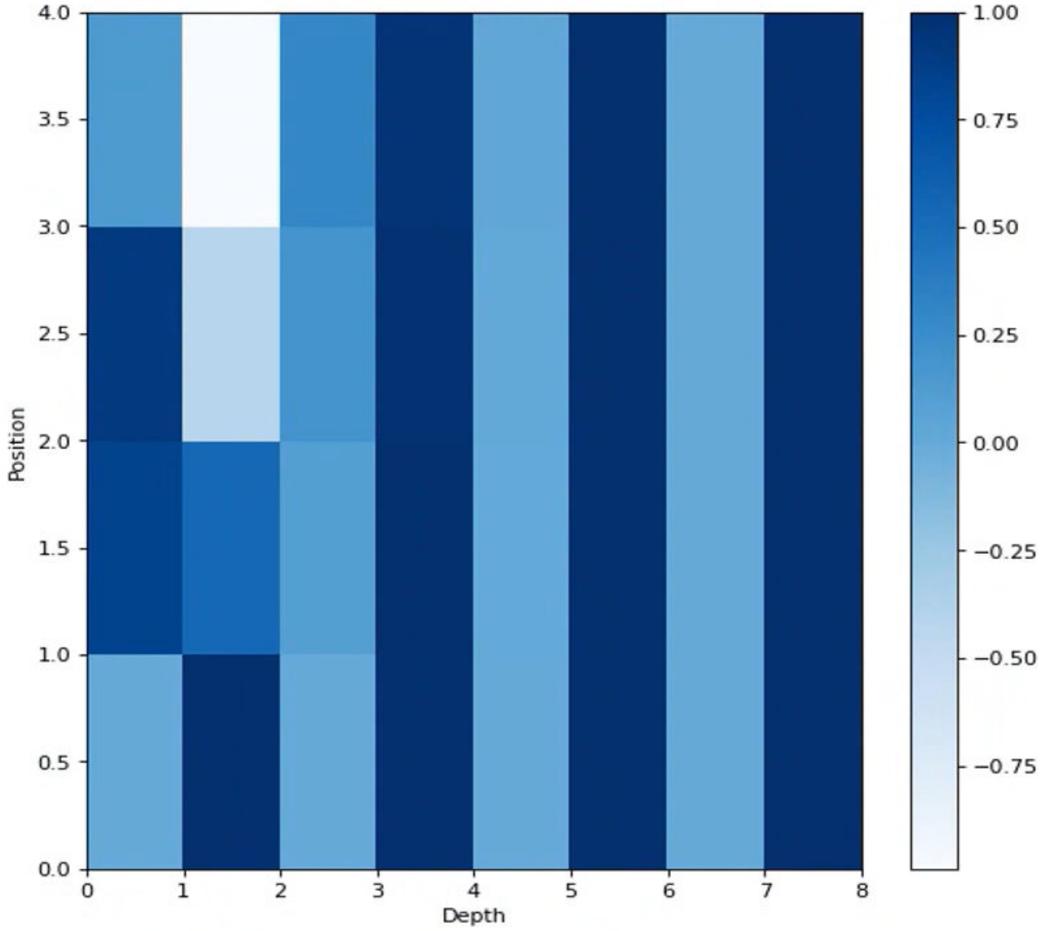


https://www.youtube.com/watch?v=4DzKM0vgG1Y

Transformer - 기계 번역의 경우

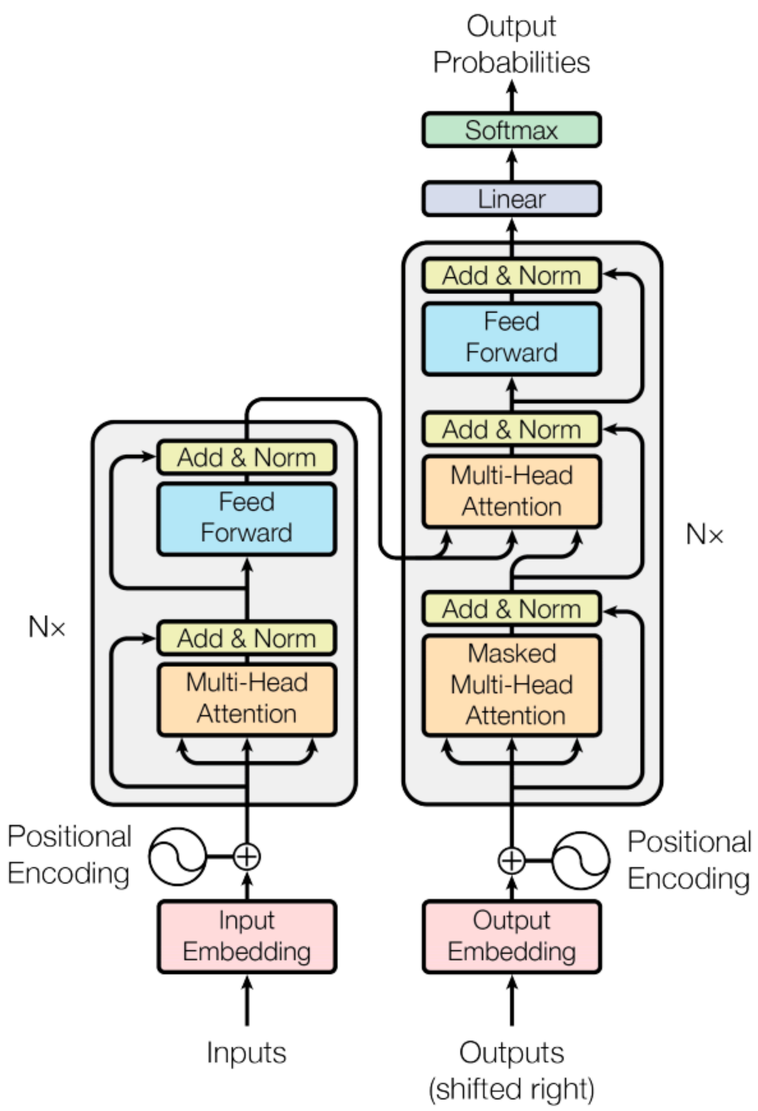
```python
import math
import matplotlib.pyplot as plt

n = 4 # 단어(word)의 개수
dim = 8 # 임베딩(embedding) 차원

def get_angles(pos, i, dim):
    angles = 1 / math.pow(10000, (2 * (i // 2)) / dim)
    return pos * angles

def get_positional_encoding(pos, i, dim):
    if i % 2 == 0: # 짝수인 경우 사인 함수
        return math.sin(get_angles(pos, i, dim))
    # 홀수인 경우 코사인 함수
    return math.cos(get_angles(pos, i, dim))

result = [[0] * dim for _ in range(n)]

for i in range(n):
    for j in range(dim):
        result[i][j] = get_positional_encoding(i, j, dim)
```

출력 결과: `plt.pcolormesh(result, cmap='Blues')`



https://www.youtube.com/watch?v=4DzKM0vgG1Y

트랜스포머 끝!!



6 Dec 2017

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

https://arxiv.org/pdf/1706.03762.pdf

https://core.today

Attention is All You Need 실습

Attention is All You Need Tutorial (German-English)

파일 수정 보기 삽입 런타임 도구 도움말

+ 코드 + 텍스트 Drive로 복사

Attention is All You Need (NIPS 2017) 실습

- 본 코드는 기본적으로 **Transformer** 논문의 내용을 최대한 따릅니다.
  - 본 논문은 **딥러닝 기반의 자연어 처리** 기법의 기본적인 구성을 이해하고 공부하는 데에 도움을 줍니다.
  - 2020년 기준 가장 뛰어난 번역 모델들은 본 논문에서 제안한 **Transformer 기반의 아키텍처**를 따르고 있습니다.
- 코드 실행 전에 **[런타임]** → **[런타임 유형 변경]** → 유형을 **GPU**로 설정합니다.

BLEU Score 계산을 위한 라이브러리 업데이트

- **[Restart Runtime]** 버튼을 눌러 런타임을 재시작할 필요가 있습니다.

```
[ ]  1 !pip install torchtext==0.6.0
```

```
Requirement already satisfied: torchtext==0.6.0 in /usr/local/lib/python3.6/dist-packages (0.6.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from torchtext==0.6.0) (1.18.5)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from torchtext==0.6.0) (2.23.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from torchtext==0.6.0) (1.15.0)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.6/dist-packages (from torchtext==0.6.0) (0.1.94)
Requirement already satisfied: torch in /usr/local/lib/python3.6/dist-packages (from torchtext==0.6.0) (1.7.0+cu101)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from torchtext==0.6.0) (4.41.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->torchtext==0.6.0) (2020.11.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->torchtext==0.6.0) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests->torchtext==0.6.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->torchtext==0.6.0) (2.10)
Requirement already satisfied: dataclasses in /usr/local/lib/python3.6/dist-packages (from torch->torchtext==0.6.0) (0.8)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.6/dist-packages (from torch->torchtext==0.6.0) (3.7.4.3)
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from torch->torchtext==0.6.0) (0.16.0)
```

데이터 전처리(Preprocessing)

- **spaCy 라이브러리**: 문장의 토큰화(tokenization), 태깅(tagging) 등의 전처리 기능을 위한 라이브러리
  - 영어(Engilsh)와 독일어(Deutsch) 전처리 모듈 설치

https://colab.research.google.com/github/ndb796/Deep-Learning-Paper-Review-and-Practice/blob/master/code_practices/Attention_is_All_You_Need_Tutorial_(German_English).ipynb

Attention is All You Need 실습

```
Epoch: 08 | Train Loss: 1.921 | Time: 698.7559933662415
```

```
epoch:  40%|■■■■      | 8/20 [1:39:53<2:34:40, 773.41s/it]
```

```
저장한 파일 이름 : ./epoch_7.pth  저장하는 데 걸린 시간 : 2.6485018730163574
```

```
training:   0%|          | 0/6250 [00:00<?, ?it/s]
training:   1%||          | 88/6250 [00:10<11:43,  8.76it/s]
training:   3%||          | 176/6250 [00:20<11:41,  8.66it/s]
training:   3%||          | 176/6250 [00:30<11:41,  8.66it/s]
training:   4%||          | 255/6250 [00:30<12:09,  8.22it/s]
training:   5%||          | 338/6250 [00:40<11:59,  8.22it/s]
training:   7%||          | 428/6250 [00:50<11:26,  8.49it/s]
training:   8%||          | 518/6250 [01:01<11:12,  8.52it/s]
training:  10%||          | 604/6250 [01:11<11:13,  8.38it/s]
```

https://github.com/CoreDotToday/DeepLearningTextBook/blob/main/NLP/Transformer/Attention_is_All_You_Need%EC%9D%98_%EC%B5%9C%EC%A2%85%EB%B3%B8.ipynb

Attention is All You Need 실습

```python
# 모델을 평가 모드로 설정
model.eval()

# 입력 문장
sent = '나'

# 문장 토큰화
proc_sent = kor_tokenizer.encode(sent)

# 후처리
post_proc_sent = postprocess(proc_sent.ids)

# Tensor로 변환 및 배치 차원 추가
input_tensor = torch.LongTensor(post_proc_sent).to(device)
input_tensor = input_tensor.unsqueeze(0)

# 출력 문장 초기화
output_tensor = torch.LongTensor([1]).to(device).unsqueeze(0)  # <sos> 토큰 + 배치 차원 추가

# 디코딩
with torch.no_grad():
    for _ in range(50):  # 최대 길이
        logits, _ = model(input_tensor, output_tensor)
        next_token = logits.argmax(-1)[:,-1]
        output_tensor = torch.cat([output_tensor, next_token.unsqueeze(-1)], dim=-1)
        if next_token.item() == 2:  # <eos> 토큰
            break

# 결과 디코딩
decoded_output = eng_tokenizer.decode(output_tensor.squeeze().tolist())
print(decoded_output)
```

```
s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s. s.
```

https://github.com/CoreDotToday/DeepLearningTextBook/blob/main/NLP/Transformer/Attention_is_All_You_Need%EC%9D%98_%EC%B5%9C%EC%A2%85%EB%B3%B8.ipynb