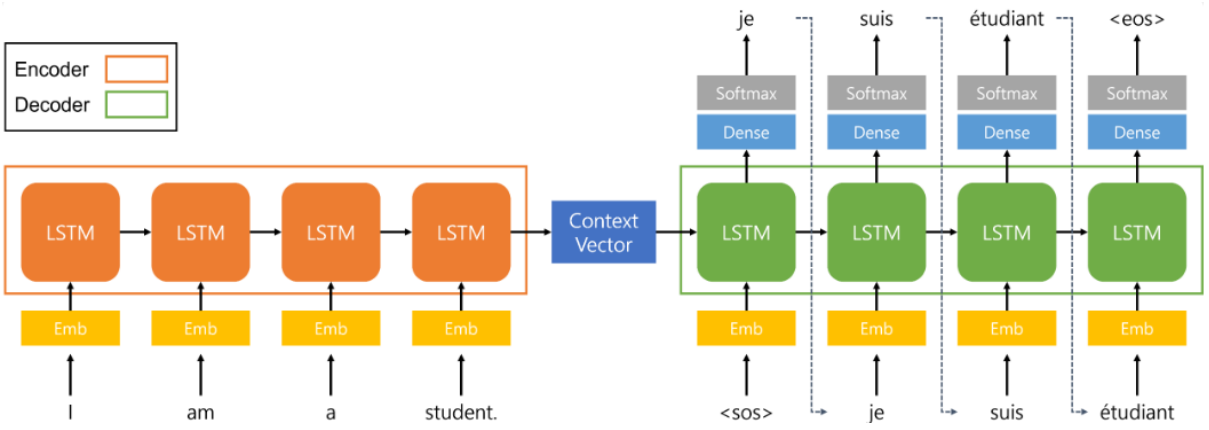
Seq2Seq with Attention

등장 배경

고정된 크기의 **Context vector**에 입력 문장의 모든 정보를 압축 → 정보 손실, 성능 저하 (**Seq2Seq** 한계점)



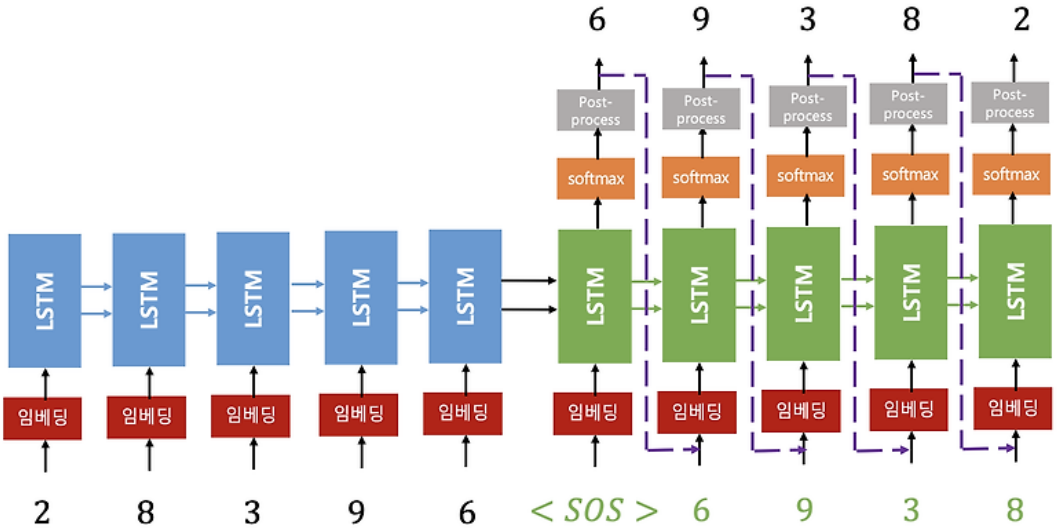
- Decoder의 각 출력시에 Context vector를 입력 → Context vector를 생성해야 한다는 한계는 동일.

- **Context vector**
  - Encoder의 입력 문장의 모든 단어들을 순차적으로 입력받고 모든 언어를 압축한 단 하나의 정보
  - float로 이루어진 하나의 벡터
  - 벡터의 크기는 모델을 처음 설정할 때 원하는 값으로 설정할 수 있으며 256, 512, 1024와 같은 숫자를 많이 사용

- **Seq2Seq**

  Seq: 시계열 데이터를 입력으로 받아서
  2: to
  Seq: 시계열 데이터를 반환하는 것

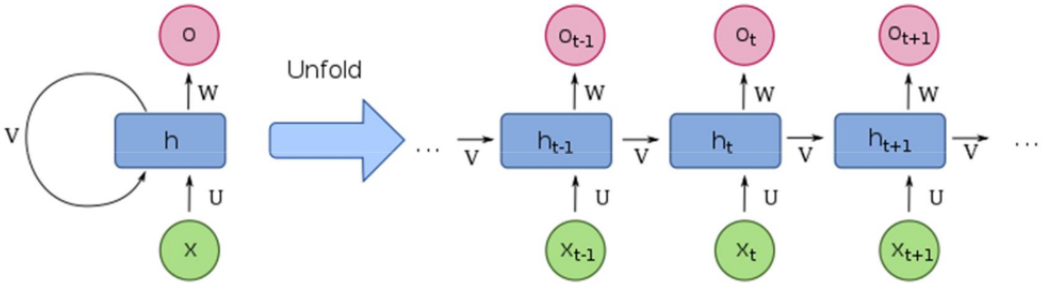
즉, 2개의 **RNN**을 연결한 모델이 **시계열 데이터**를 입력으로 받아 시계열 데이터를 반환할 때 Seq2Seq라고 함



- **RNN**

  Recurrent Neural Networks

  - 순환신경망은 입력층에서 출력층으로의 입력값 전달과 동시에 은닉층의 정보가 다음 은닉층으 로 이어지는 구조를 가진 신경망
  - 이러한 구조를 통해 순차 데이터나 시계열 데이터와 같이 일정한 순서를 가지는 데이터나 일정 시간 간격으로 배치된 데이터 수열을 처리할 수 있음



- **시계열 데이터**
  - 시계열 데이터(Time-series data)란, 시간 순서에 따라 관측된 데이터를 의미합니다.
  - 주식 가격, 기상 정보, 웹사이트의 사용자 트래픽 등 다양한 분야에서 사용됩니다.
  - 시계열 데이터의 주요 특징은 시간적 순서를 따르는 점과 연속성이 있습니다.

Attention이 사용된 이유

Encoder를 통해 각 토큰(문장)의 정보를 **고정된 길이**의 Context vector로 압축할 때 정보의 손실이 발생한다. 또한 손실이 발생한 **고정된 길이**의 Context vector를 이용해 Decoder에서 각 셀의 값을 생성한 것이 충분한 정보를 통해 토큰(단어)를 생성하는 것이 아니다. 이를 방지하지 위해 Attention 알고리즘이 고안되었다.

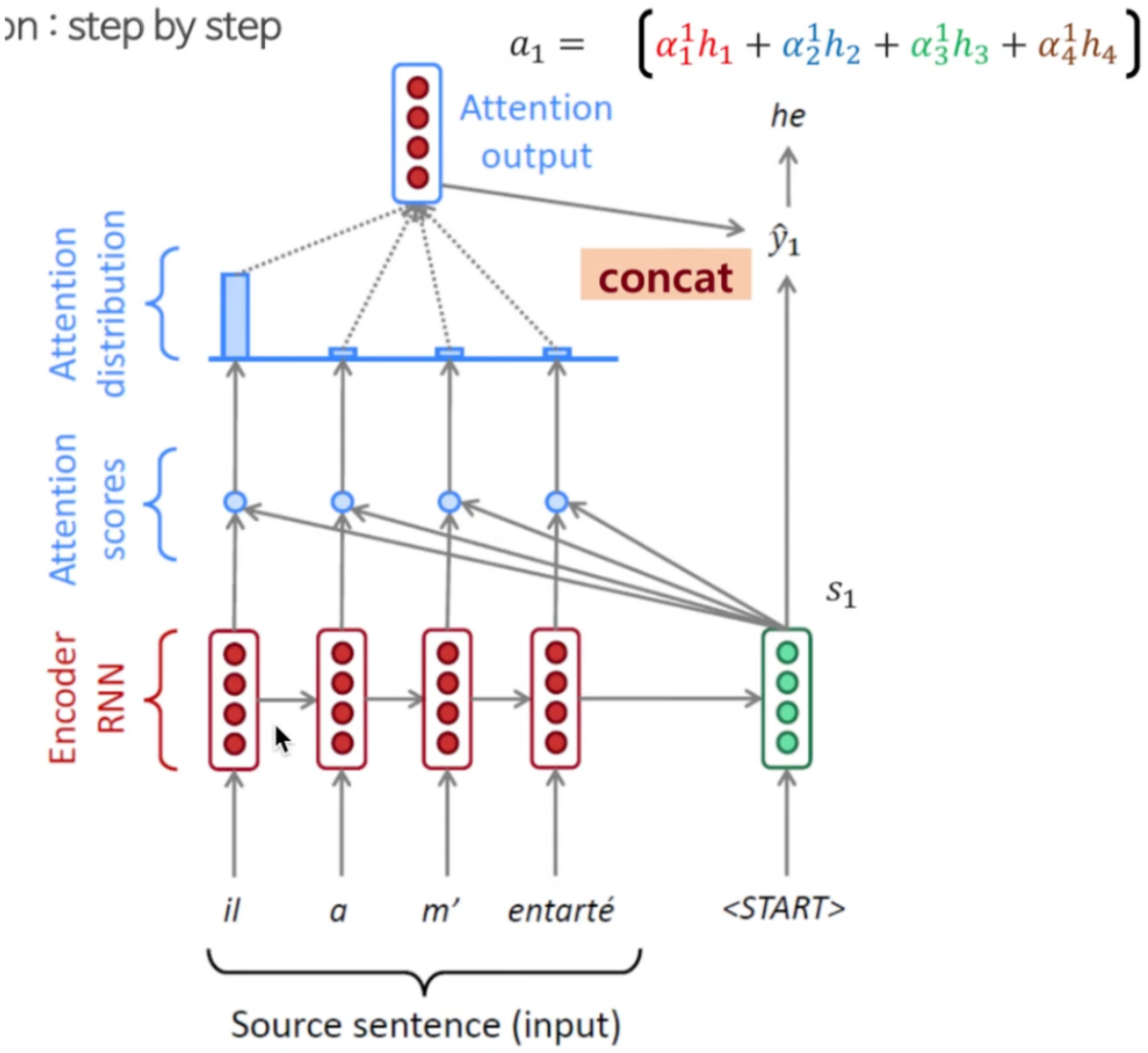
Attention의 아이디어

Decoder에서 각 단어를 생성할 때, Context vector뿐만 아니라 은닉층의 모든 hidden state값과 현재 Decoder 셀에서 생성되는 단어와의 관계를 고려하자.

기본 아이디어

Decoder에서 각 단어를 생성할 때, Context vector v 뿐만 아니라 은닉층의 모든 hidden state 값 (h_1, h_2, ..., h_n)과 현재 Decoder에서 생성되는 단어와의 관계를 고려하자.

동작 원리



1. **Attention score** 계산 → **Energy** 값 도출
2. **Attention 분포** 도출 → **Attention 가중치** 도출
3. **Attention value** 도출
4. Decoder의 hidden state에 Attention value를 결합 → Dense Layer 입력값 구성.
    a. 단순히 결합(concat) 하는 방법
    b. Decoder의 t번째 hidden state에 더함. 차원이 일치할 경우에만 사용 가능.
5. Dense Layer 통과 이후 Softmax를 이용해 출력 단어 생성.

- **Attention score**

  현재 Decoder의 t 시점에서 단어를 예측하기 위해, Encoder의 모든 hidden state 값이 Decoder의 t 시점 hidden state와 얼마나 관련있는지 구한 점수

💡 **일반적으로 Seq2Seq의 hidden state는 RNN 레이어의 hidden state를 의미한다.**

- n개의 은닉 상태 → n개의 Attention score
- Encoder의 hidden state에 따른 Attention score의 결합 = **Energy** 값

Attention score 계산 방법

- Dot product in Attention score
- Scaled dot product in Attention score

- **Energy**

  Seq2Seq with Attention 기법의 Encoder에서 모든 hidden state에 대한 Attention scor]의 집합 벡터.

  예를 들어 Encoder의 i 번째 hidden state에 대한 Attention value가 s_i로 정의될 때 Decoder의 t 번째 Energy 값은 다음 식으로 정의된다.

$$e_t = [s_1, s_2, \ldots, s_n]$$

  - e_t : Decoder의 t번째 Energy값.
  - n: Encoder의 hidden state 개수.

- **Attention 분포**

Seq2Seq with Attention에서 Attention score의 집합인 Energy 값에 Softmax를 적용해 얻은 확률 분포.

$$\alpha^t = \text{softmax}(e^t)$$

- α^t: Decoder의 t 시점에서의 Attention 분포.
- e^t: Decoder의 t 시점에서의 Attention score.
- Attention 분포를 통해 얻은 각각의 값을 Attention 가중치 라고 부름.

- Encoder의 hidden state가 Decoder의 t 시점에 영향을 주는 정도을 알 수 있음.
- 각 Query(Decoder의 t 시점)에 대해 Key(Enoder의 i번째 hidden state)의 점수(Energy 값)이 정규화됨 ⇒ 특정 Key가 Query와 얼마나 잘 매칭되는지를 반영하는 가중치로 변환.

- **Attention 가중치**
  - Attention 분포의 각각의 값.
  - 특정 Key(Encoder의 i번째 hidden state값)가 Query(Decoder의 t번째 hidden state값)와 얼마나 잘 매칭되는지를 반영
  - Attention 가중치의 값 ⬆ → 해당 Key가 Query에 영향을 많이 준다.
- **Attention value**

현재 Decoder의 시점에 대한 입력 시퀀스의 가중표현.

Seq2Seq with Attention에서 Attention 가중치에 Encoder의 hidden state의 가중 합한 것. 수식으로 다음으로 정의됨.

$$a^t = \sum_{i=1}^{N} \alpha_i^t h_i$$

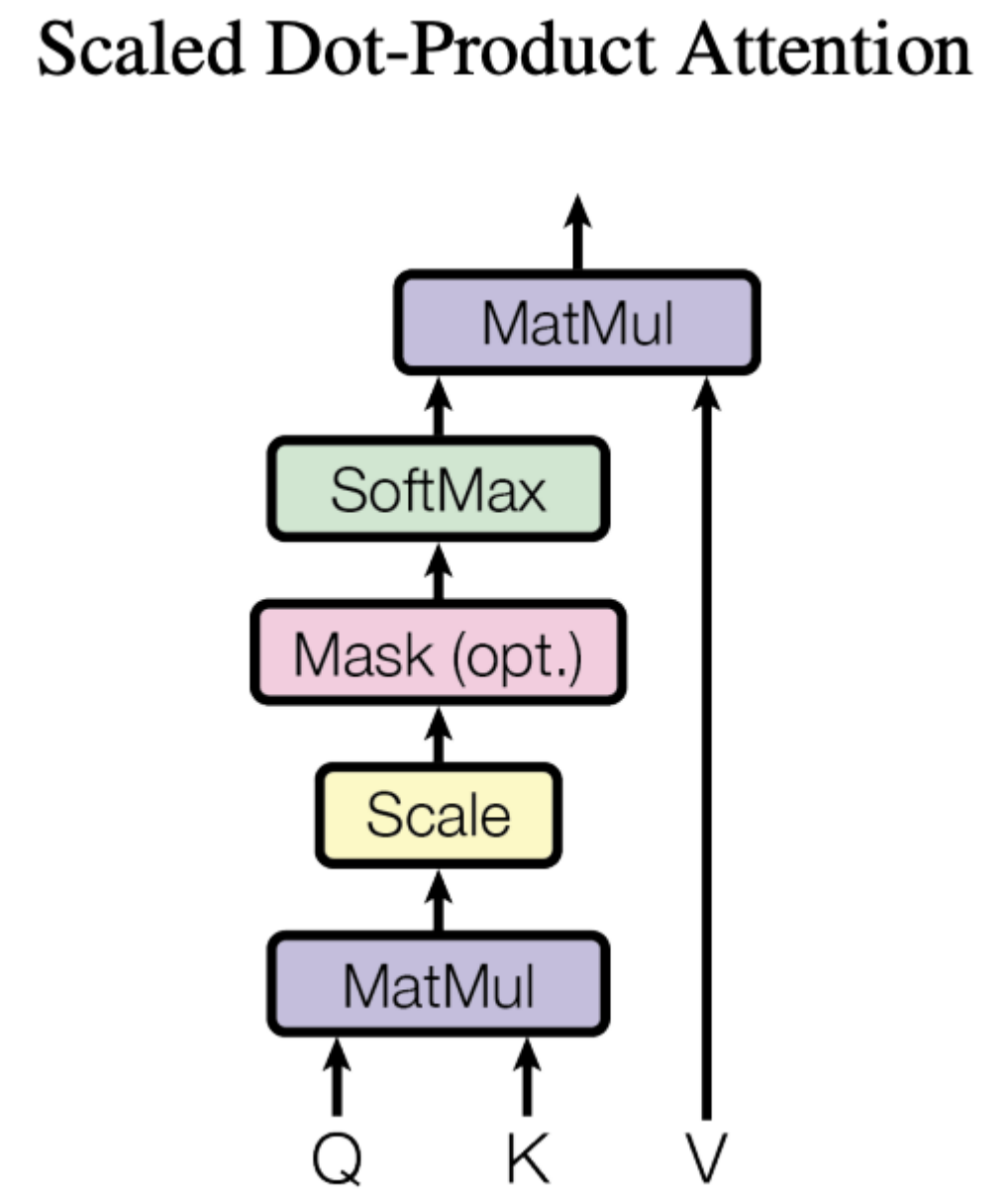
- t: Decoder의 현재 출력 인덱스.
- i: Encoder의 hidden state 번호.
- N: Encoder의 hidden state 개수.
- a^t: t번째 Attenion value 값.
- α_i^t : Decoder의 t번째 Attention 분포의 i번째 값. 즉 i 번째 Attention 가중치.
- h_i: Encoder의 i 번째 hidden state 값.

> ⓘ **위 과정에서 h_i는 Attention 연산의 Value 역할을 수행함.**

Masks는 어느 문서에 넣어야 되는지 고민.

- Attention value는 Context vector와 같이 사용 가능.

Scaled Dot-product Attention 전체 흐름도(1,2,3 과정).



수식 정리

Attention score 계산을 **dot product** 이용할 경우 Attention value는 다음과 같이 정의된다.

$$\text{Attention}(Q_t, K, V) = \text{Softmax}(\frac{Q_t K^T}{\sqrt{d_k}})V$$

$$\text{Output}_t = \text{Dense}(\text{Concat}(\text{Attention}(Q_t, K, V), H_t))$$

- t: Decoder의 현재 시점
- Q_t: Decoder의 t 시점 hidden state
- K: Encoder의 Hidden state값.
- V: Encoder의 hidden state값.

> ⚠ **Warning**
>
> Seq2Seq에서 Key와 Value는 동일하지만 Attention 메커니즘에서 다른 값이 될 수도 있다.

- **dot product** (Scaled dot product in Attention score)

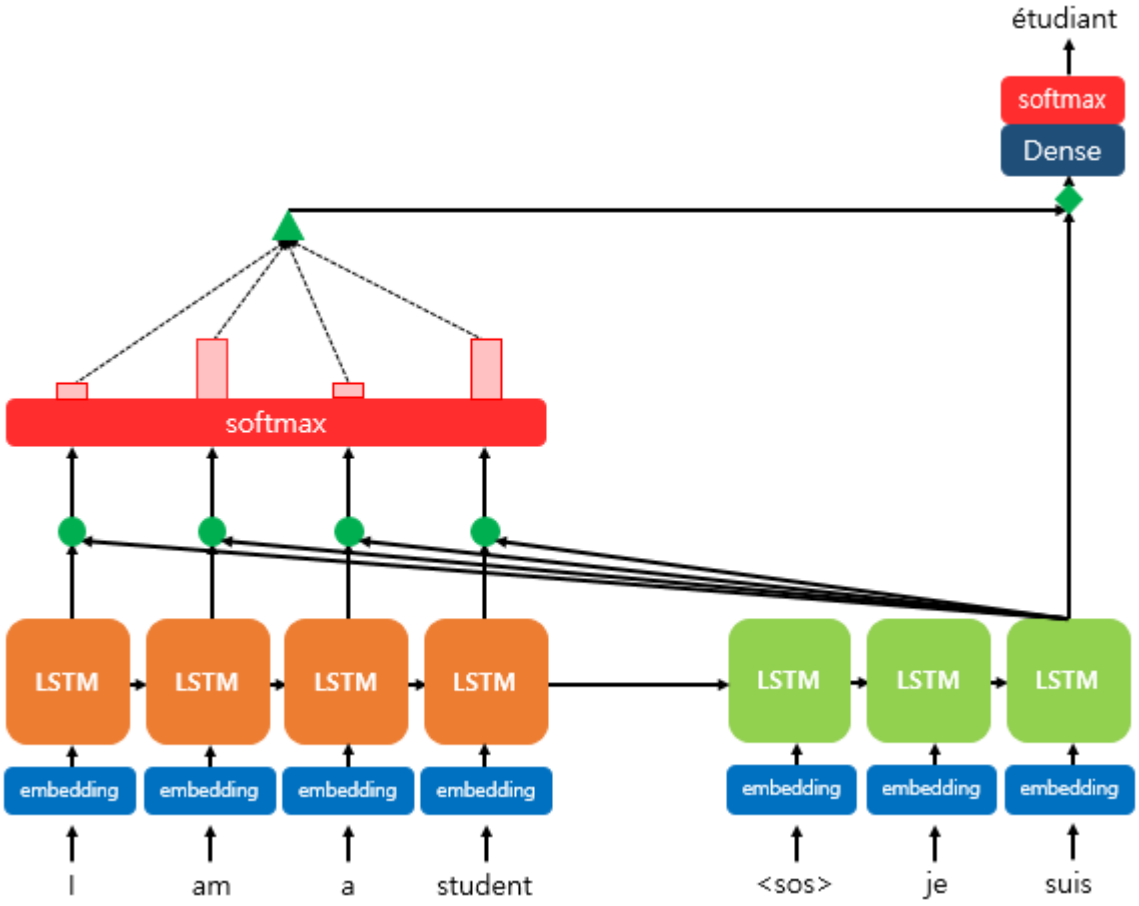
  Dot product in Attention score에 스케일 팩터를 추가한 방법으로 아래 식으로 정의됨.

  > *현재 Decoder의 시점이 t이며 Encoder의 i번째 hidden state에 대한 Attention score를 구한다고 가정.*

  $$\text{Score}(Q_t, K_i) = \frac{Q_t K_i^T}{\sqrt{d_k}}$$

  - Q_t는 Decoder의 t번째 hidden state 벡터. → Attention의 Query에 해당
  - K_i^T는 Encoder의 i번째 hidden state 벡터의 전치. → Attention의 Key에 해당.
  - $\sqrt{d_k}$는 K의 차원 수에 루트를 씌운 값으로 스케일링 목적으로 사용.
  - Q_t 는 $1 \times d_k$ 차원이며 K^T는 $d_k \times n$ ⇒ Attention score는 $1 \times n$ 차원.(n은 Key 벡터의 개수)

Decoder의 동작 순서에 대해 다시 정리해보자



1. Decoder의 t-1 시점의 출력(Dense 레이어 이후의 Softmax를 통해 예측된 단어)과 t-1 시점의 hidden state를 통해 t 시점의 RNN(LSTM)셀의 hidden state 계산.
2. t 시점의 hidden state 값을 통해 t 시점의 Attention value 계산.
3. t 시점의 RNN hidden state와 t 시점의 Attention value를 결합하여 Dense layer의 입력 벡터로 사용 → Softmax를 통해 다음 단어 예측.