Seq2Seq에서 이전 모든 data를 특정 크기의 dimension을 갖는 vector로 embedding이 거의 불가능하다.

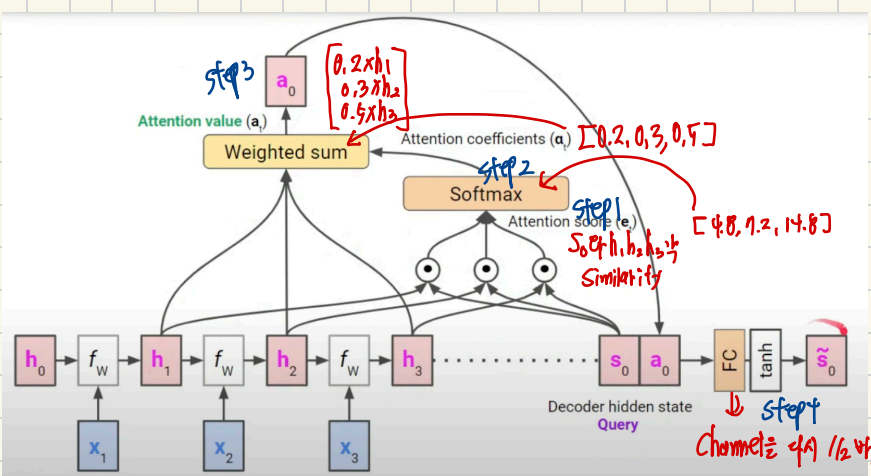
↓

decoding시 encoder의 최종 vector만 사용하지 말고, encoder 중간 중간의 output도 참조하자.

〈Attention Idea〉 어디에 집중할지를 결정한다.

Q, K를 사용하여 weight (가중치)를 생성하고 Value를 곱해주는 연산.

Q. : decoder의 time t의 현재상태. (기준)
K. : encoder의 hidden state들 } 현재 time의 encoder에 집중!
V. : 〃



위의 과정을 모든 S_n에 대해 진행한다.

$h_1 h_2 \cdots h_n \in R^h$

$s_t \in R^h$ # 현재 state.

$e_t = [s_t^T h_1, s_t^T h_2, \cdots, s_t^T h_T]$ # encoder의 hidden state 들과 유사도 계산

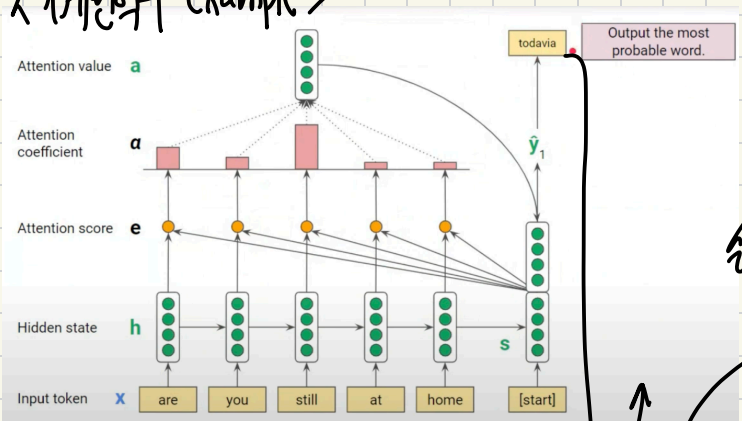
$\alpha_t = \text{softmax}(e_t) \in R^T$ # softmax

$a_t = \sum_{t=1}^{T} [\alpha_t]_i \cdot h_i \in R^h$ # value를 곱해서

$[a_t, s_t] \in R^{2h}$ # state랑 append 한다.

similarity는 cosine sim이 간단해보여서 성능이 좋아 자주 쓴다.

<어텐션의 example>



실제로는 max length 설정.

while result = <eos> 인 경우

피드백 (training에는 teacher force)

〈Transformers〉

MLP와 CNN은 본질적 동일하다. ⇒ $\hat{y} = W_2 \sigma(W_1 x)$ 결국 x들의 weighted sum을 사용한 모델

RNN ⇒ 역시 결국 $\tanh(W_2 \cdot \tanh(W_1 \cdot x_1))$ 하면 x들의 weighted sum.

x나온 data들 속의 각 x들끼리 서로 상호작용하여 x를 표현한다. ⇒ "Self attention" : x를 표현시 x뿐만 아니라 주변 data들도 모두 고려하여 x를 표현하라.

$x \rightarrow$ Linear Transform

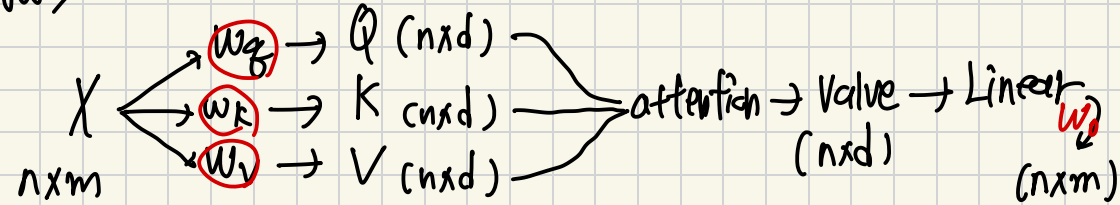
- $W_Q \rightarrow Q$
- $W_K \rightarrow K$
- $W_V \rightarrow V$

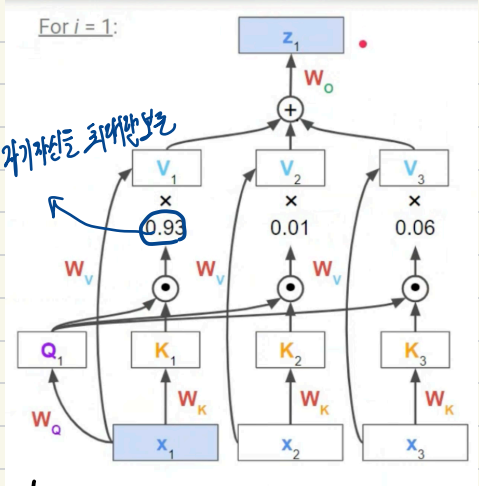
Seq2Seq와 다르게 Q, K, V를 Linear Transform으로 직접 만든다.

※ W_Q, W_K, W_V은 모든 input에 same 하다

learnable parameter가서 어떻게 Q3, K3, V3 역할을 잘 수행하는지를 학습한다.

〈flow〉



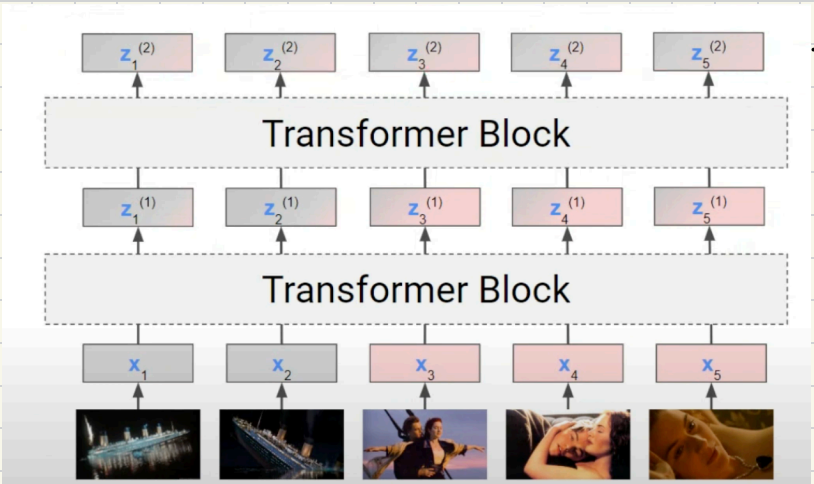


↳ "X_1을 X_2, X_3와의 상호작용으로 새롭게 표현해서 z_1이 된다"

z_1은 X_1의 상호작용이된 새로운 자신이 된다.

즉, 모든 X들은 주변 data들과의 상호작용으로 재표현 (Transform) 함. ☆

"contextualize"



즉, Self Attention은 자기자신을 문맥에 맞게 재표현하는 방식이다.