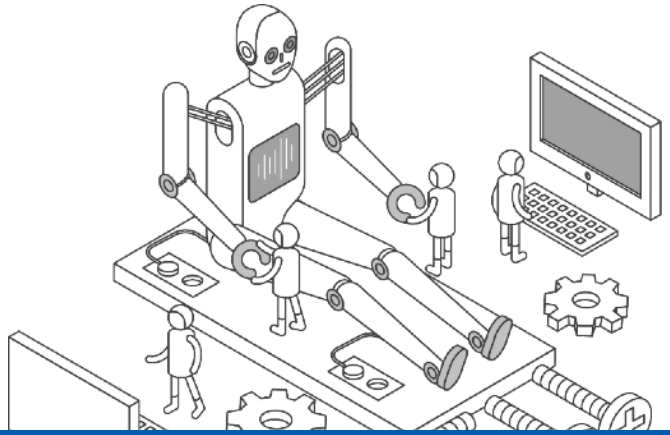
2021 직업계고 AI 전문교육

ARTIFICIAL INTELLIGENCE
BIG DATA
SMART FACTORY

AI·빅데이터 심화과정

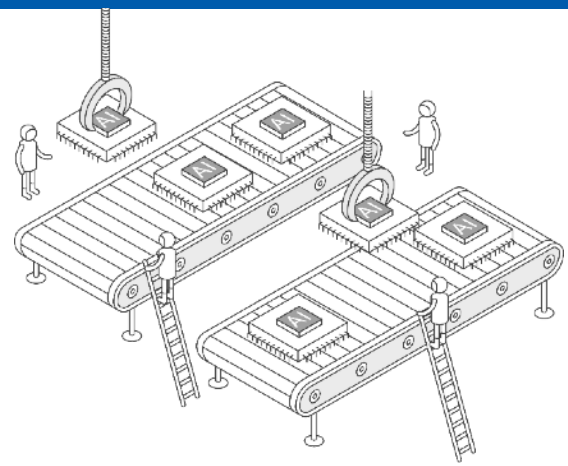
26. 뉴스 요약봇 만들기

박수철
github.com/scpark20
GaudioLab, 모두의연구소



서울특별시교육청 kocoafab (주)네패스 코코아랩 모두의 연구소

26. 뉴스 요약봇 만들기



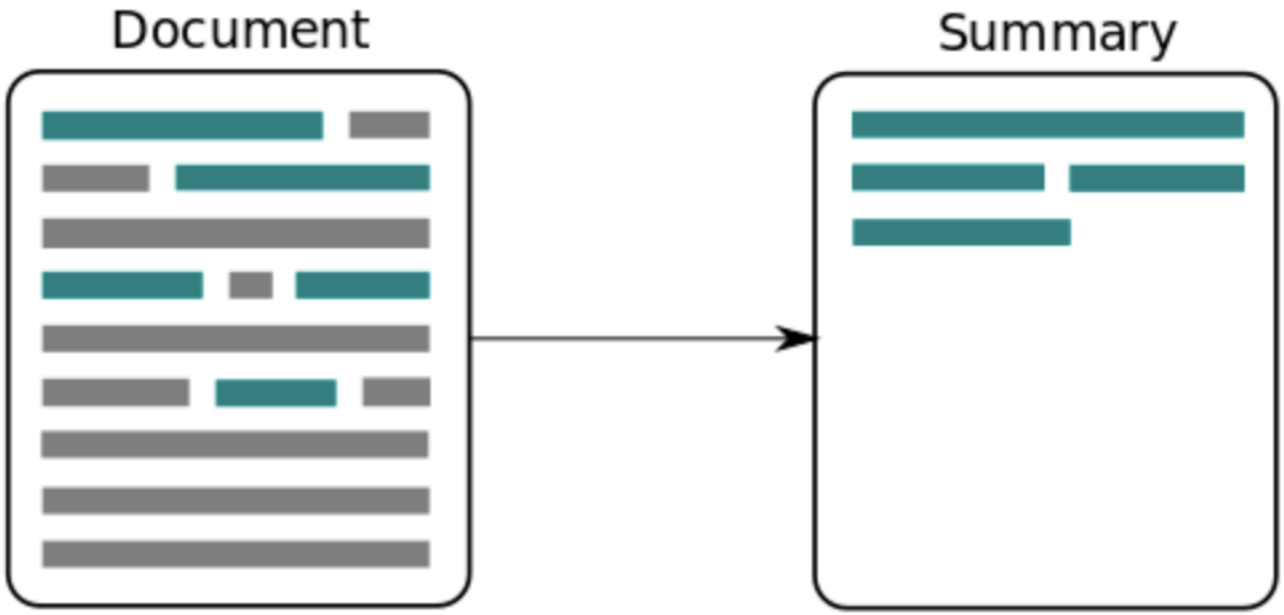
Seq2Seq
9/24 - RNN으로 소설쓰기 (Aiffel 외)
9/29, 10/1 - 26. 뉴스 요약봇 만들기
10/6, 10/8 - 27. 트랜스포머로 만드는 대화형 챗봇

CNN/GAN
10/13, 10/15 - 22. 난 스케치를 할테니 너는 채색을 하거라
10/20, 10/22 - 21. 흐린 사진을 선명하게
10/27, 10/29 - 18. GO/STOP!

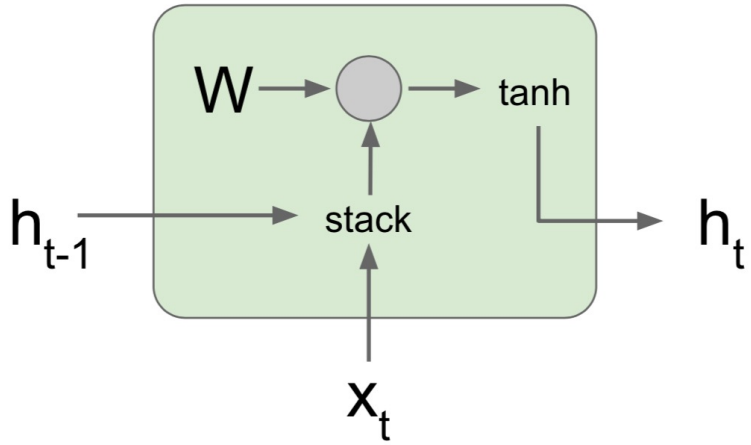
RNN+CNN
11/3, 11/5 - RNN으로 음성인식하기 (Aiffel 외)
11.10, 11/12 - 19. 직접 만들어보는 OCR

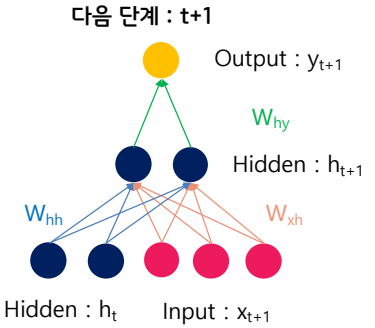
Ablation study
11/17 - 17. 없다면 어떻게 될까?

Seq2Seq
9/24 - RNN으로 소설쓰기 (Aiffel 외)
9/29, 10/1 - 26. 뉴스 요약봇 만들기
10/6, 10/8 - 27. 트랜스포머로 만드는 대화형 챗봇



Vanilla RNN Gradient Flow





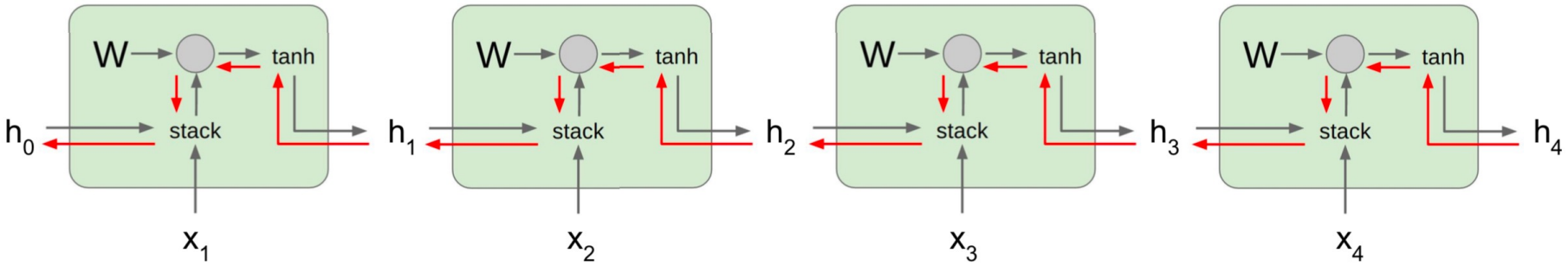
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Vanilla RNN Gradient Flow

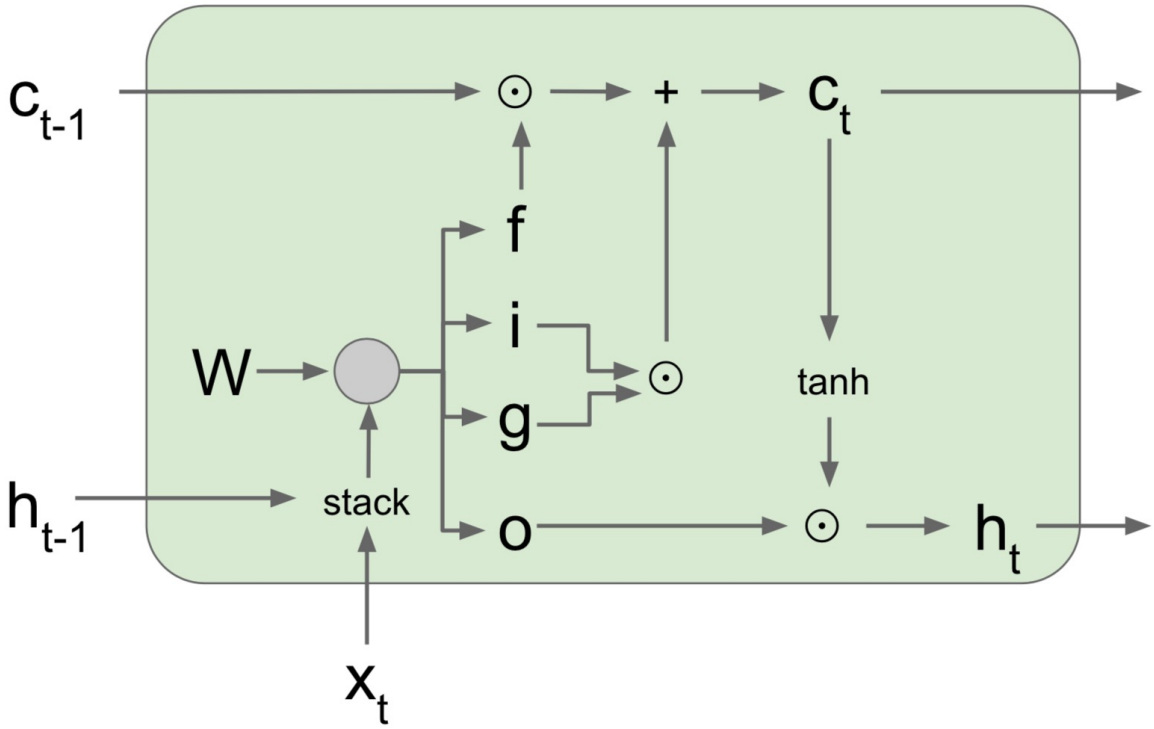
Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



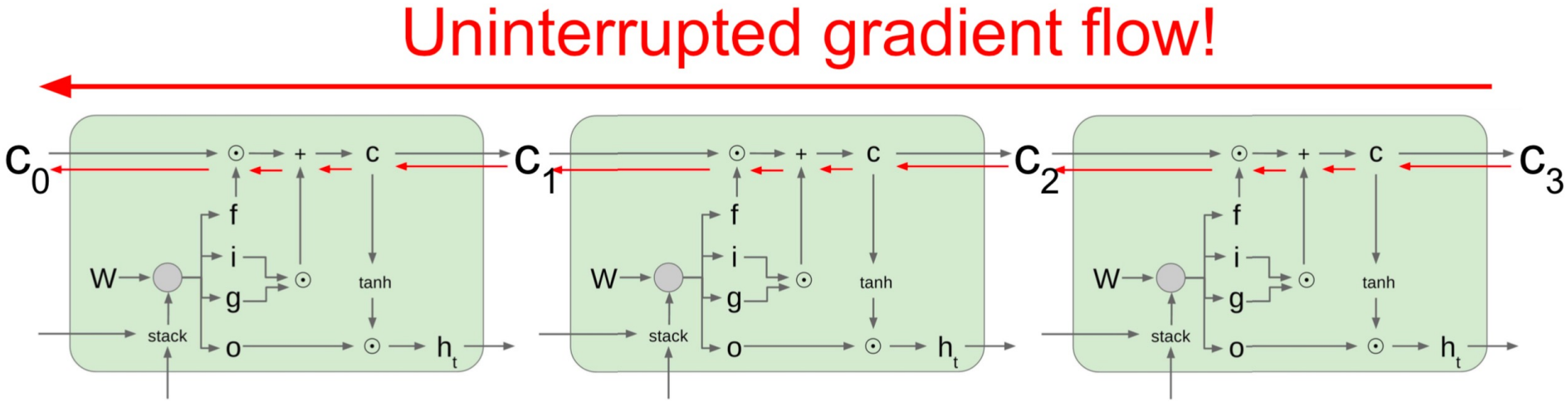
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

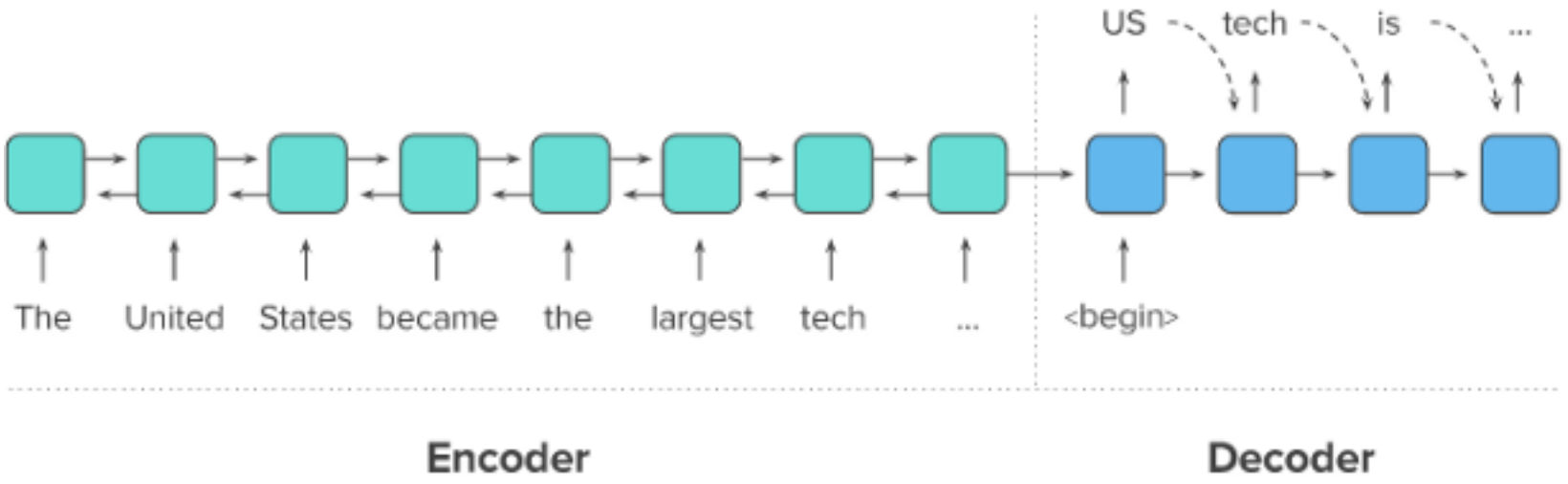
$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



- Seq2Seq는 Sequence to Sequence의 약어로 주어진 시퀀스를 조건(condition)으로 하여 새로운 시퀀스를 만들어내는 작업을 말합니다.
- 시퀀스는 주로 문자로 이루어진 문장을 말하지만, 음성 데이터, 한달간의 날씨와 같은 데이터도 뜻합니다.
- 번역작업(프랑스어 문장-영어 문장), 챗봇(물음-대답), TTS(문장-음성), 날씨 예측(과거 날씨-미래 날씨) 등의 예를 들 수 있습니다.



https://medium.com/dl-for-product-and-service/abstractive-text-summary-with-reinforcement-learning-ab2458ab29d5

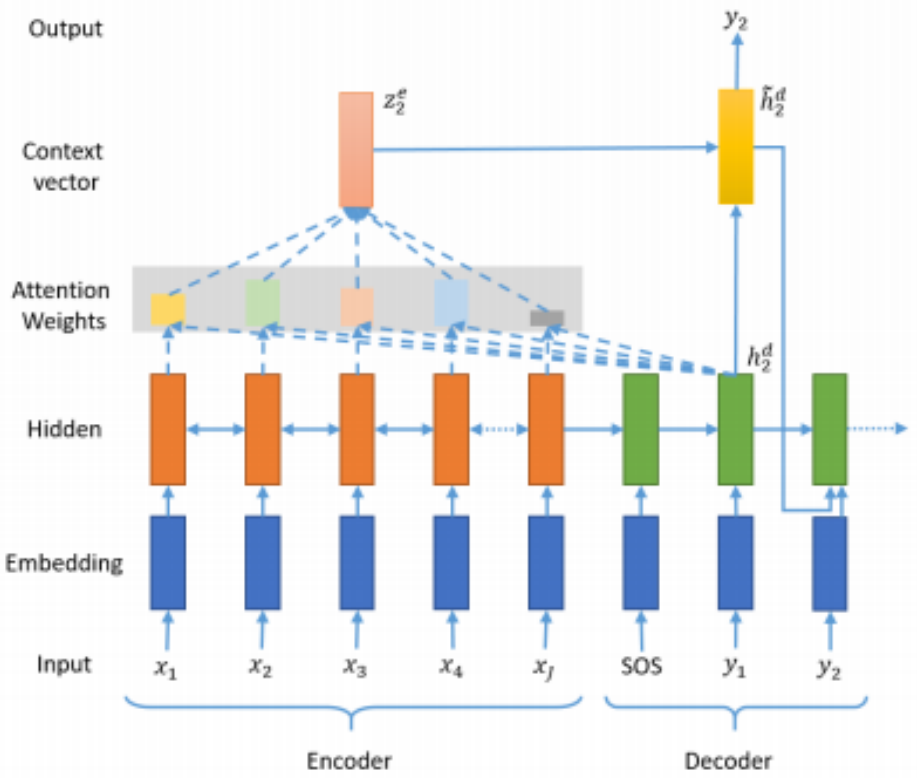
- Encoder에 의해 encoding된 정보를 효과적으로 decoder에서 사용할 수 있도록 attention이라는 메카니즘을 도입합니다.

- Attention은 다음과 같이 작동합니다.

  1. decoder의 한 step의 hidden vector와 encoder의 모든 hidden vector들 간에 어떠한 연산을 수행하여 attention weights를 만듭니다.

  2. Attention weights를 비율로 하여 encoder의 hidden vector들을 weighted sum하여 context vector를 만들어 냅니다.

  3. Context vector를 decoder의 hidden vector와 concat하여 최종 output을 하기 위해 사용합니다.



https://arxiv.org/pdf/1812.02303.pdf

Attention을 하기 위한 energy e_{ij}는 다음과 같이 계산합니다.

$$e_{ij} = a\left(s_{i-1}, h_j\right) = v_a^{\top} \tanh\left(W_a s_{i-1} + U_a h_j\right)$$

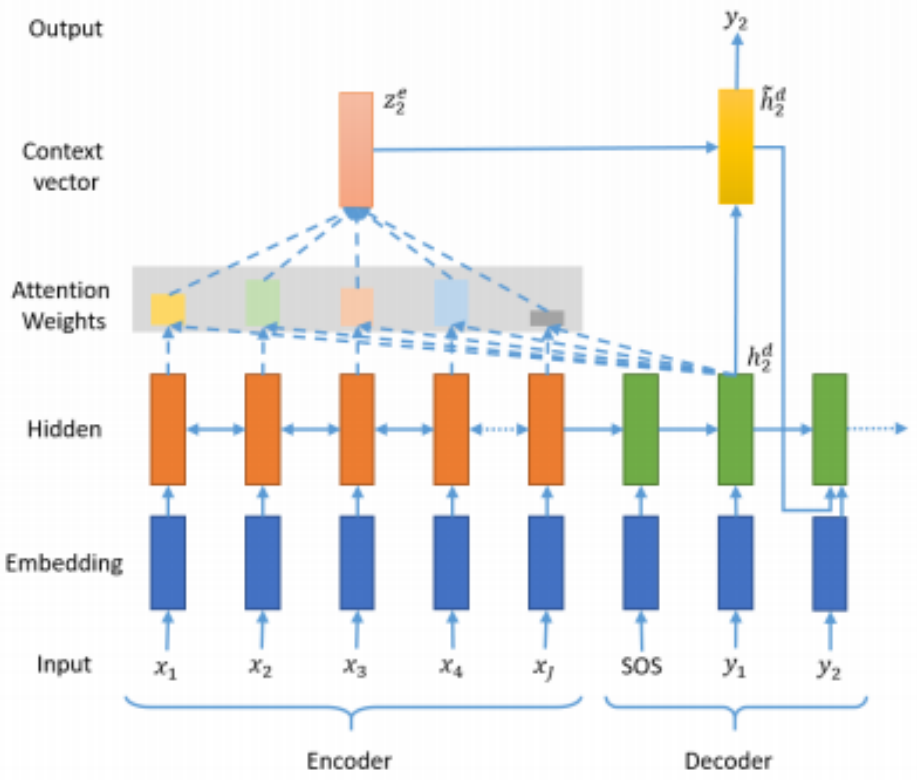
s_{i-1}: i-1시점의 decoder hidden state
h_j: j시점의 encoder hidden state
v_a^T, W_a, U_a: trainable parameters

Attention weights는 energy e_{ij}를 softmax 연산을하여 확률분포의 형태를 만들어 구합니다.

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_k \exp\left(e_{ik}\right)}$$

마지막으로 context vector는 energy e_{ij}와 encoder hidden state들을 weighted sum하여 구합니다.

$$c_i = \sum_j \alpha_{ij} h_j$$



https://arxiv.org/pdf/1812.02303.pdf

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).


똑똑
파이팅!!