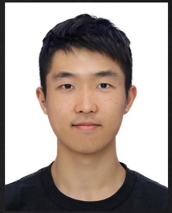
DL Chatbot seminar
Day 03

Seq2Seq / Attention

HELLO!

I am Jaemin Cho

- Vision & Learning Lab @ SNU
- NLP / ML / Generative Model
- Looking for Ph.D. / Research programs



You can find me at:

- heythisischo@gmail.com
- j-min
- J-min Cho
- Jaemin Cho

Today We will cover

- ✘ RNN Encoder-Decoder for Sequence Generation (Seq2Seq)
- ✘ Advanced Seq2Seq Architectures
- ✘ Attention Mechanism
  - PyTorch Demo
- ✘ Advanced Attention architectures

1.

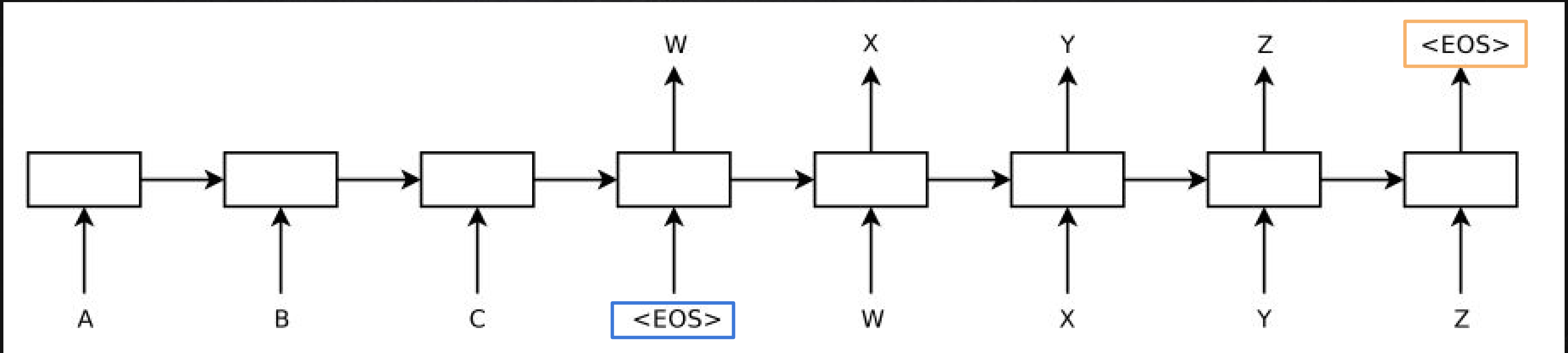
RNN Encoder-Decoder for Sequence Generation

RNN Encoder-Decoder
Neural Conversation Model
Alternative Objective: MMI

Sequence-to-Sequence

- Goal
  - Source Sentence를 받고 Target Sentence를 출력

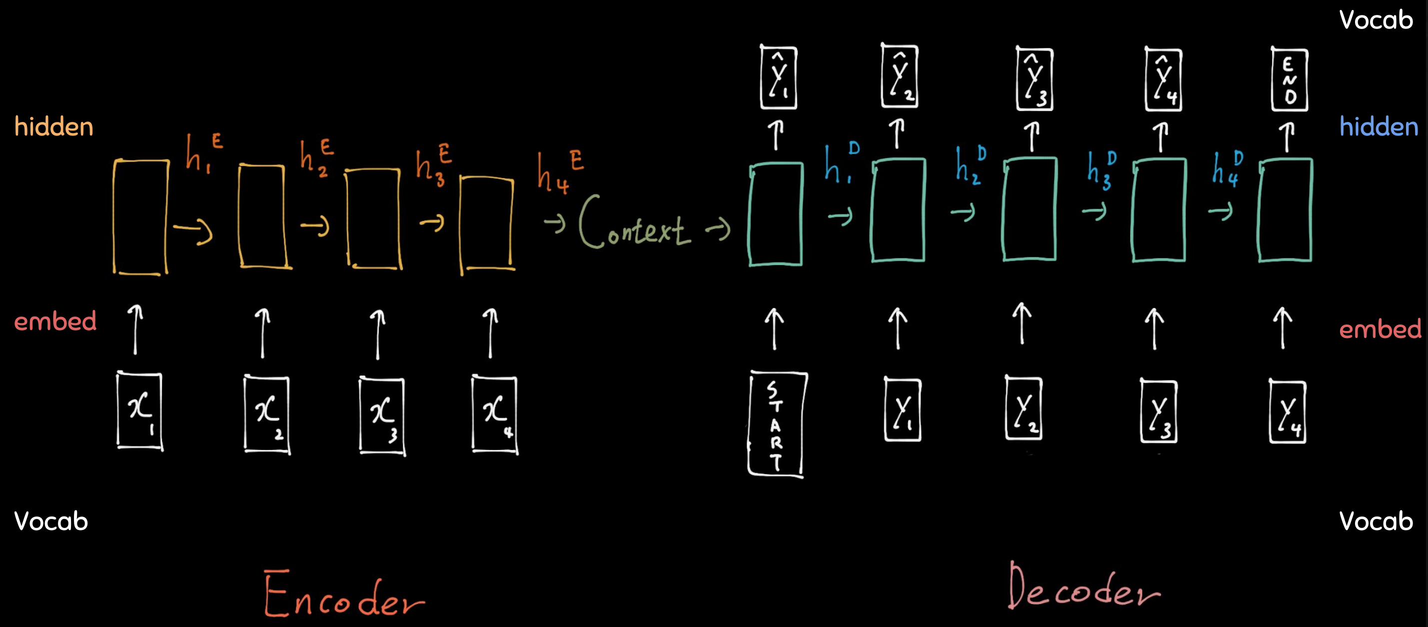
이제 디코딩 그만!



이제 Source Sentence입력이 끝났다!
여기 뒤부터는 Target Sentence!

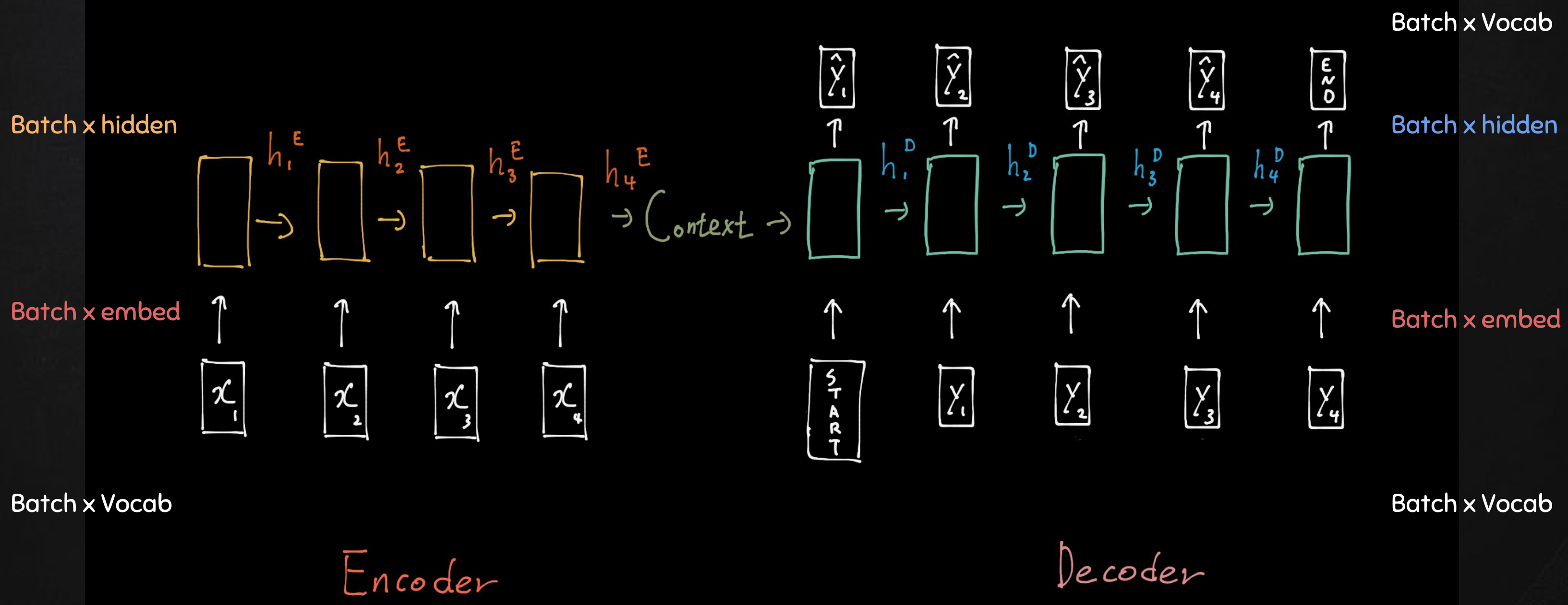
"Sequence to Sequence Learning with Neural Networks" (2014)

RNN Encoder - Decoder (Seq 2 Seq)



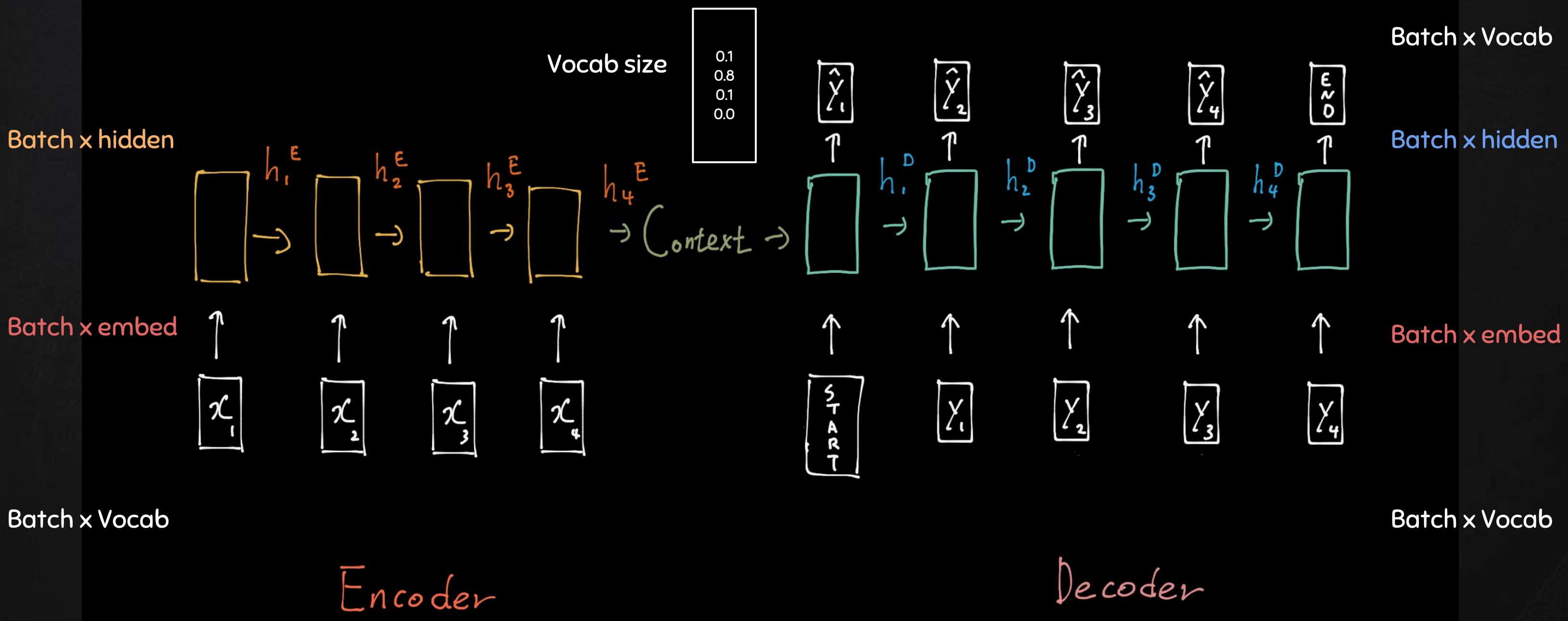
"Sequence to Sequence Learning with Neural Networks" (2014)

RNN Encoder - Decoder (Seq2Seq)



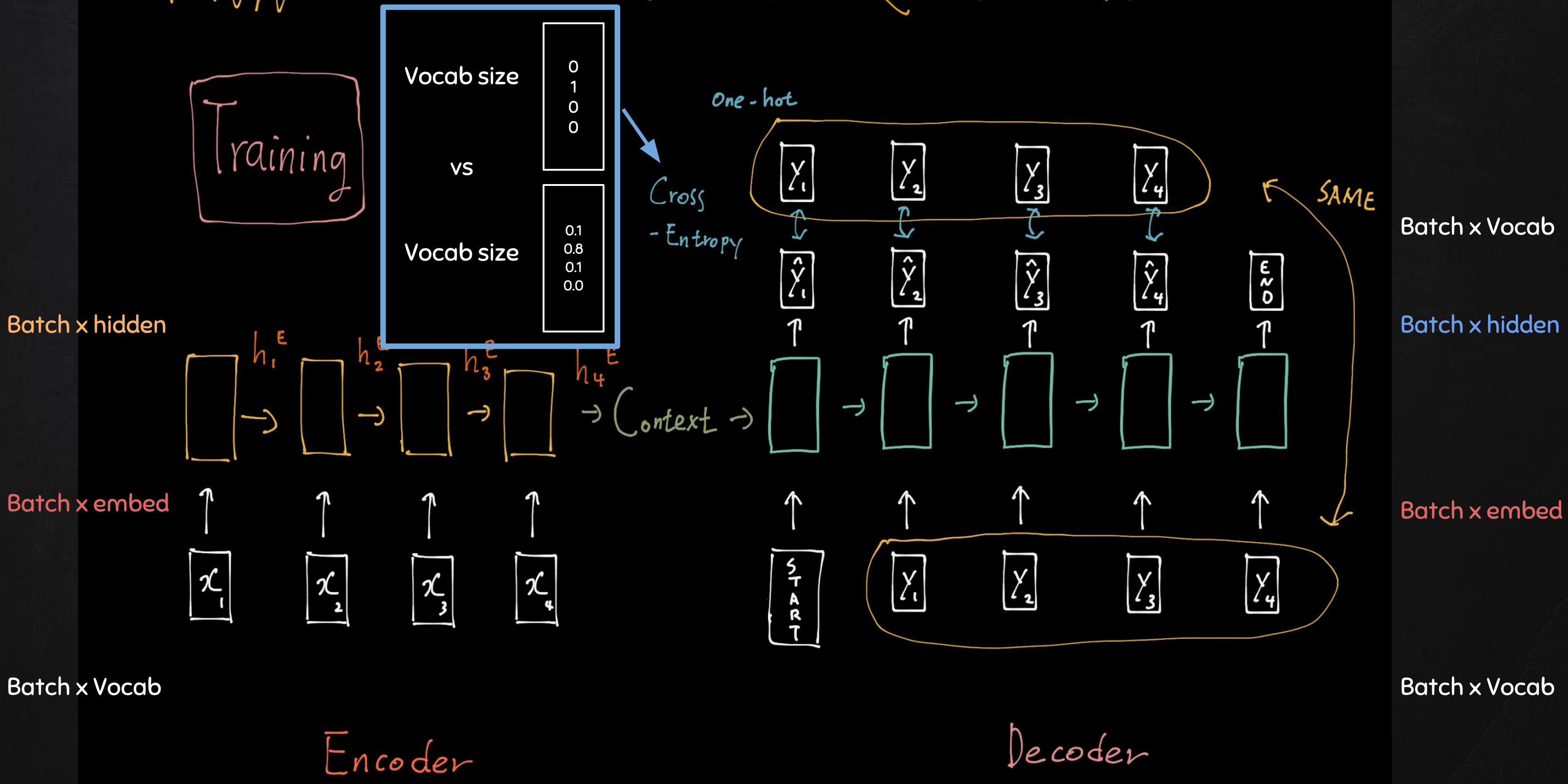
"Sequence to Sequence Learning with Neural Networks" (2014)

RNN Encoder - Decoder (Seq 2 Seq)



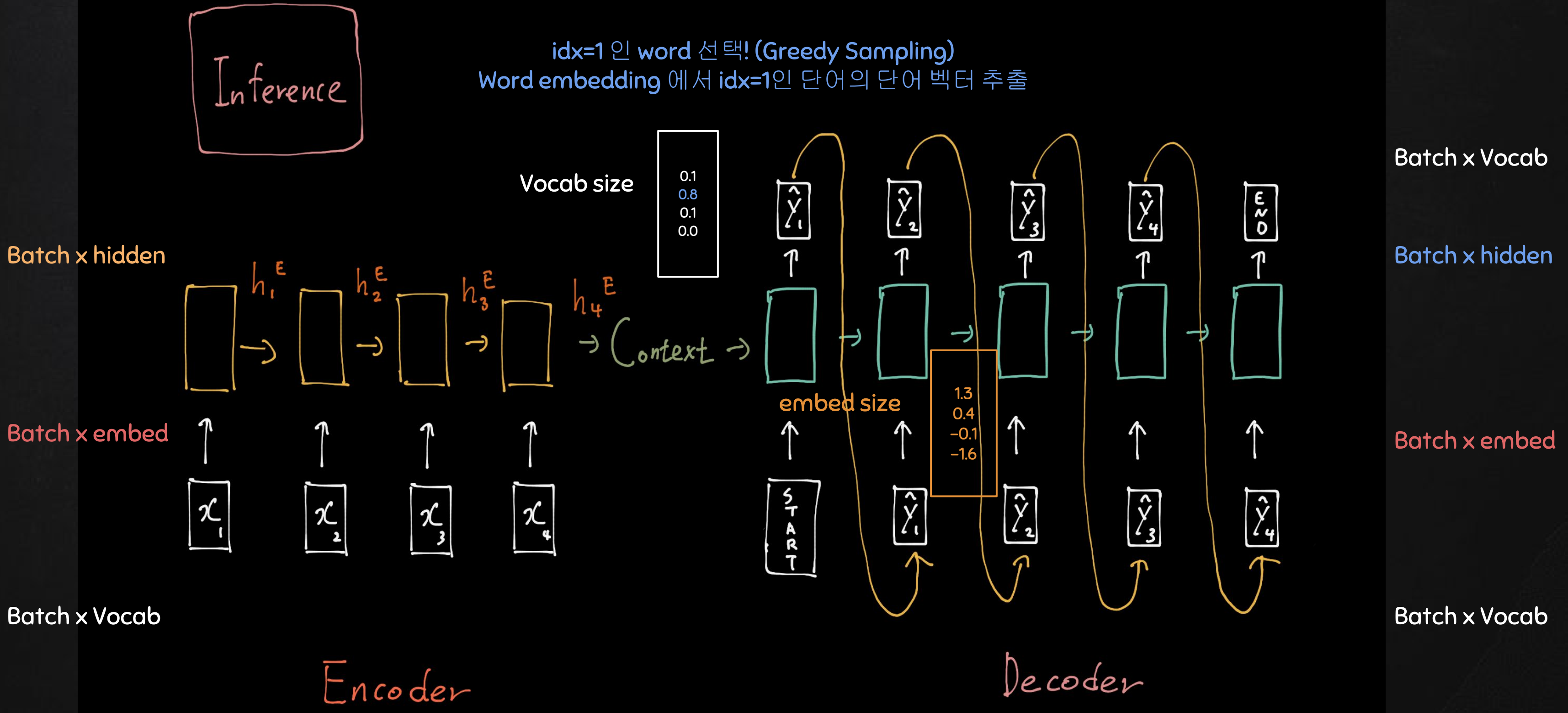
"Sequence to Sequence Learning with Neural Networks" (2014)

RNN Encoder - Decoder (Seq 2 Seq)



"Sequence to Sequence Learning with Neural Networks" (2014)

RNN Encoder - Decoder (Seq2Seq)



"Sequence to Sequence Learning with Neural Networks" (2014)

Neural Conversation Model

IT Helpdesk Troubleshooting dataset

- ✘ First Seq2Seq Chatbot

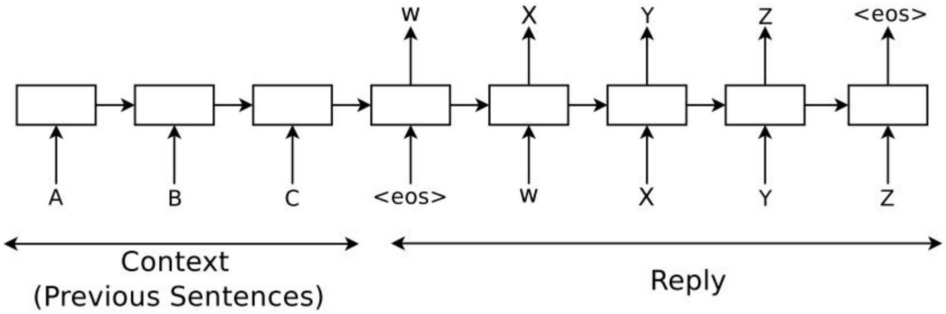


Figure 1. Using the *seq2seq* framework for modeling conversations.

Machine: *hi*
Human: *hello*
Machine: *could you please let me know what are the operating systems you are using ?*
Human: *i am using linux*
Machine: *ok it was in the ssl vpn on which you stated it was broken*
Machine: *please check if you are logged in to corp network*
Machine: *what is the error that you are running please*
Human: *i am seeing an error related to vpn*
Machine: *what is the error message that you are getting when connecting to vpn using network connect ?*
Human: *connection refused or something like that*
Machine: *may i know the version of network connect you connect ?*
Human: *i am not sure i know that*

"A Neural Conversational Model" (2015)

Drawbacks of Seq2Seq for Chatbot

- ✘ Too generic Response
- ✘ Source constraint on generation process
  - Only source of variation is at the output
- ✘ No persona
- ✘ Cannot capture 'higher-level' representations

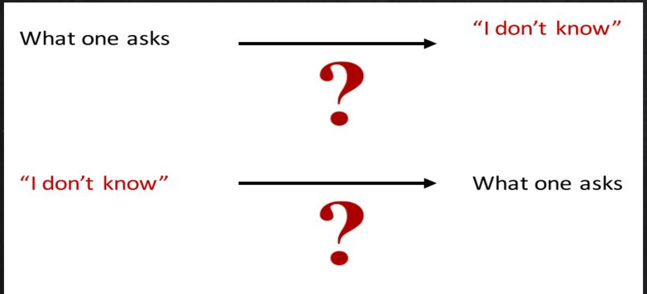
Too Generic Response

- Standard Seq2Seq Objective
  - Maximize log-likelihood (MLE)

문맥 파악 x
그냥 많이 나오는 단어를 많이 말하게 됨

$$\hat{T} = \arg\max_{T} \left\{ \log p(T|S) \right\}$$

- Only selects for targets given sources, not the converse



- Does not capture actual objective in human communication

Input: What are you doing?			
−0.86	I don't know.	−1.09	Get out of here.
−1.03	I don't know!	−1.09	I'm going home.
−1.06	Nothing.	−1.09	Oh my god!
−1.09	Get out of the way.	−1.10	I'm talking to you.
Input: what is your name?			
−0.91	I don't know.	...	
−0.92	I don't know!	−1.55	My name is Robert.
−0.92	I don't know, sir.	−1.58	My name is John.
−0.97	Oh, my god!	−1.59	My name's John.
Input: How old are you?			
−0.79	I don't know.	...	
−1.06	I'm fine.	−1.64	Twenty-five.
−1.17	I'm all right.	−1.66	Five.
−1.17	I'm not sure.	−1.71	Eight.

Table 1: Responses generated by a 4-layer SEQ2SEQ neural model trained on 20 million conversation pairs take from the OpenSubtitles dataset. Decoding is implemented with beam size set to 200. The top examples are the responses with the highest average probability log-likelihoods in the N-best list. Lower-ranked, less-generic responses were manually chosen.

"A Diversity-Promoting Objective Function for Neural Conversation Models" (2015)

Maximum Mutual Information (MMI)

- Alternative objective
  - Maximum Mutual Information (MMI)

$$\log \frac{p(S,T)}{p(S)p(T)}$$

$$\hat{T} = \arg\max_{T} \left\{ \log p(T|S) - \boxed{\log p(T)} \right\}$$

Prior 가 높은 (자주 나오는) 표현들에 penalty 부과

Avoid too generic response

- With Hyperparameter λ

$$\hat{T} = \arg\max_{T} \left\{ \log p(T|S) - \lambda \boxed{\log p(T)} \right\}$$

=> Anti-Language Model (MMI-antiLM)

$$= \log p(T|S) + \log p(S) - \log p(S|T)$$

- With Bayes Theorem
  - Weighted MMI is rewritten as below

$$\begin{aligned} &= \arg\max_{T} \big\{ (1-\lambda) \log p(T|S) \\ &\qquad + \lambda \log p(S|T) - \lambda \log p(S) \big\} \\ &= \arg\max_{T} \big\{ (1-\lambda) \log p(T|S) + \lambda \log p(S|T) \big\} \end{aligned}$$

=> MMI-bidi

"A Diversity-Promoting Objective Function for Neural Conversation Models" (2015)

Maximum Mutual Information (MMI)

- ✘ Generated more diverse response

Model	BLEU	*distinct-1*	*distinct-2*
SEQ2SEQ	1.28	0.0056	0.0136
MMI-antiLM	1.74 (+35.9%)	0.0184 (+228%)	0.066 (407%)
MMI-bidi	1.44 (+28.2%)	0.0103 (+83.9%)	0.0303 (+122%)

Table 3: Performance of the SEQ2SEQ baseline and two MMI models on the OpenSubtitles dataset.

Input: What are you doing?			
−0.86	I don't know.	−1.09	Get out of here.
−1.03	I don't know!	−1.09	I'm going home.
−1.06	Nothing.	−1.09	Oh my god!
−1.09	Get out of the way.	−1.10	I'm talking to you.
Input: what is your name?			
−0.91	I don't know.	...	
−0.92	I don't know!	−1.55	My name is Robert.
−0.92	I don't know, sir.	−1.58	My name is John.
−0.97	Oh, my god!	−1.59	My name's John.
Input: How old are you?			
−0.79	I don't know.	...	
−1.06	I'm fine.	−1.64	Twenty-five.
−1.17	I'm all right.	−1.66	Five.
−1.17	I'm not sure.	−1.71	Eight.

Input: What are you doing?	
1. I've been looking for you.	4. I told you to shut up.
2. I want to talk to you.	5. Get out of here.
3. Just making sure you're OK.	6. I'm looking for a doctor.
Input: What is your name?	
1. Blue!	4. Daniel.
2. Peter.	5. My name is John.
3. Tyler.	6. My name is Robert.
Input: How old are you?	
1. Twenty-eight.	4. Five.
2. Twenty-four.	5. 15.
3. Long.	6. Eight.

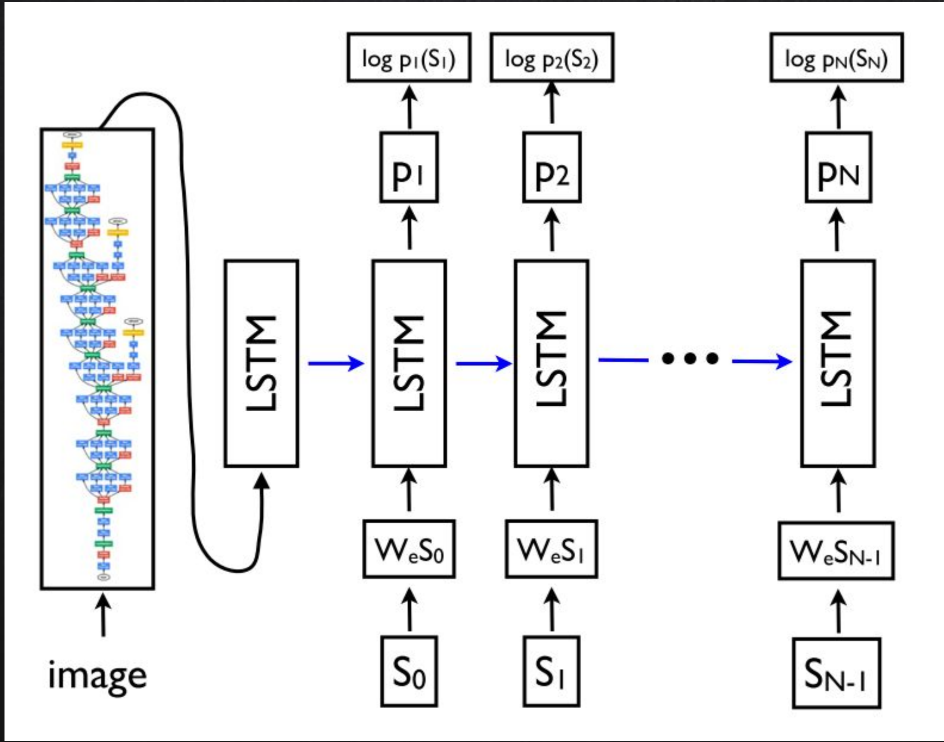
"A Diversity-Promoting Objective Function for Neural Conversation Models" (2015)

2.

Advanced Seq2Seq Architectures

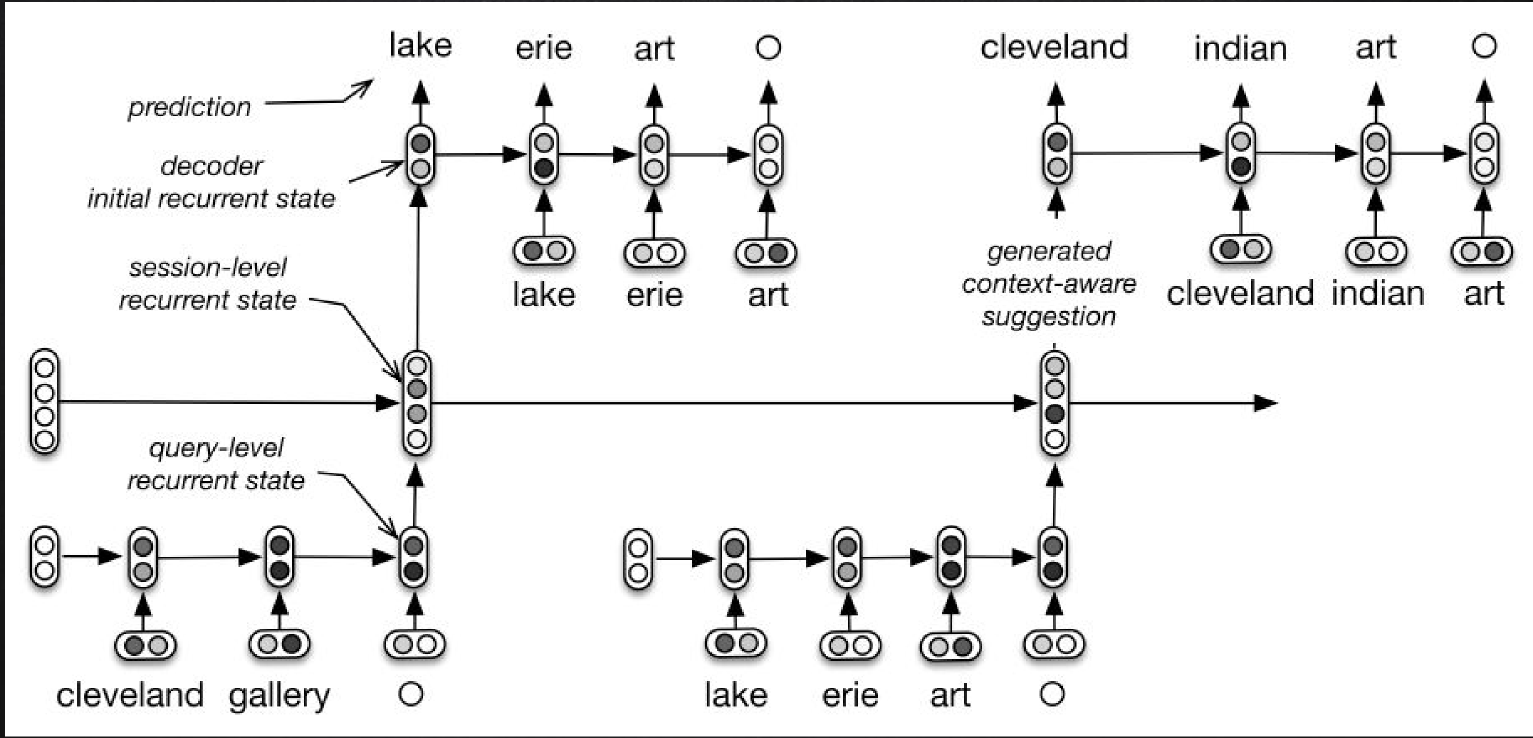
Image Captioning: Show and Tell
Hierarchical Seq2Seq: HRED / VHRED
Personalized embedding: Persona-Based Neural Conversation Model
Learning in Translation: CoVe

Show And Tell



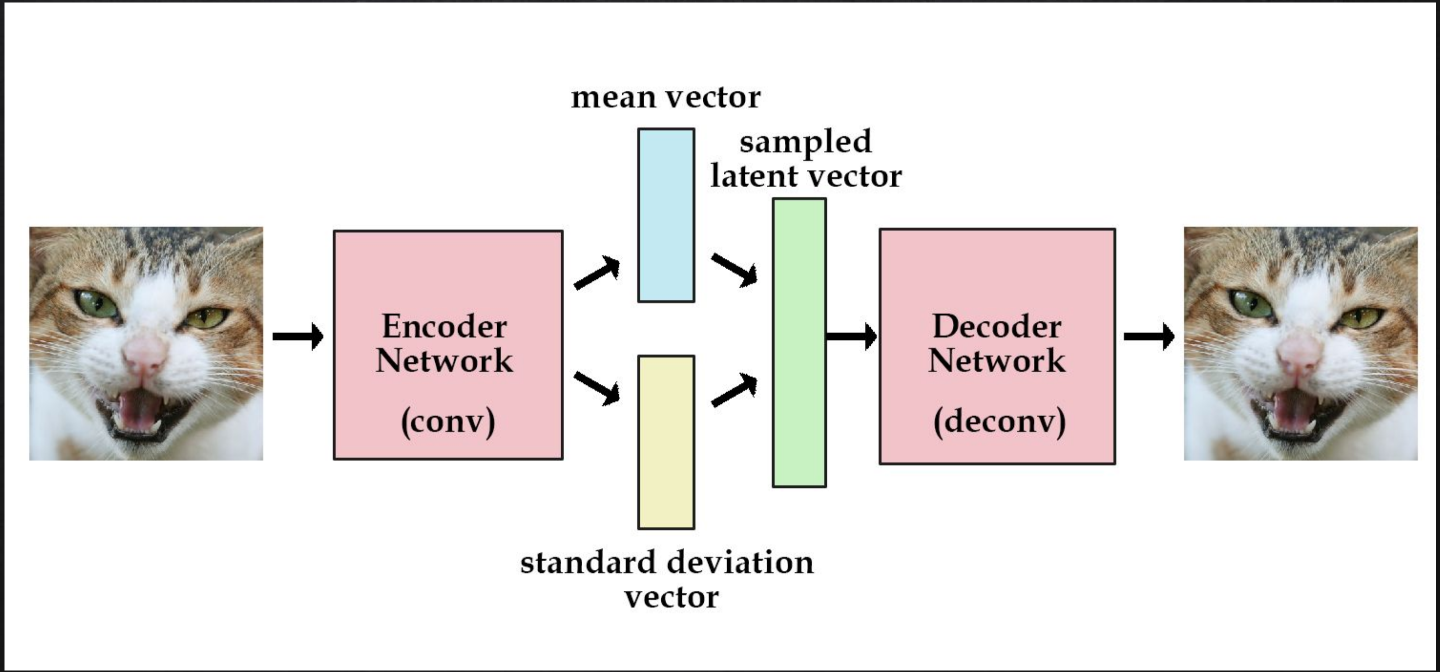
"Show and Tell: A Neural Image Caption Generator" (2015)

Hierarchical Recurrent Encoder-Decoder (HRED)



"A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion" (2015)

Variational Autoencoder (VAE)



“Auto-Encoding Varational Bayes” (2014)

Latent Variable HRED (VHRED)

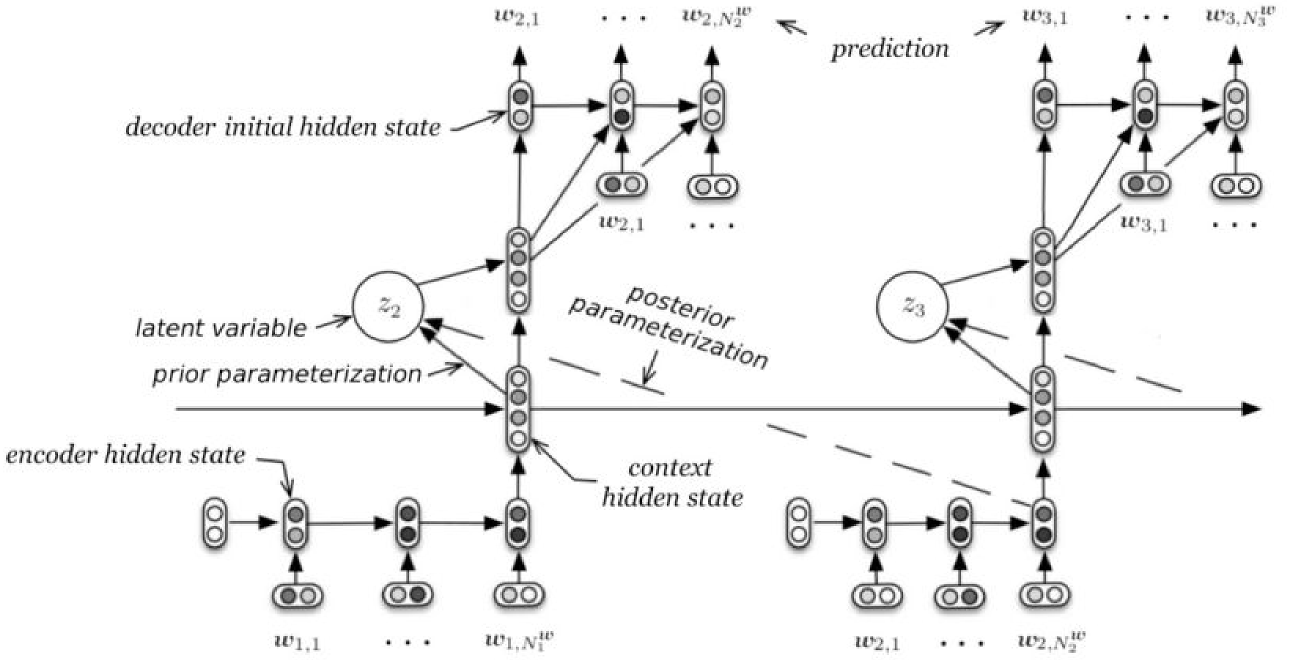
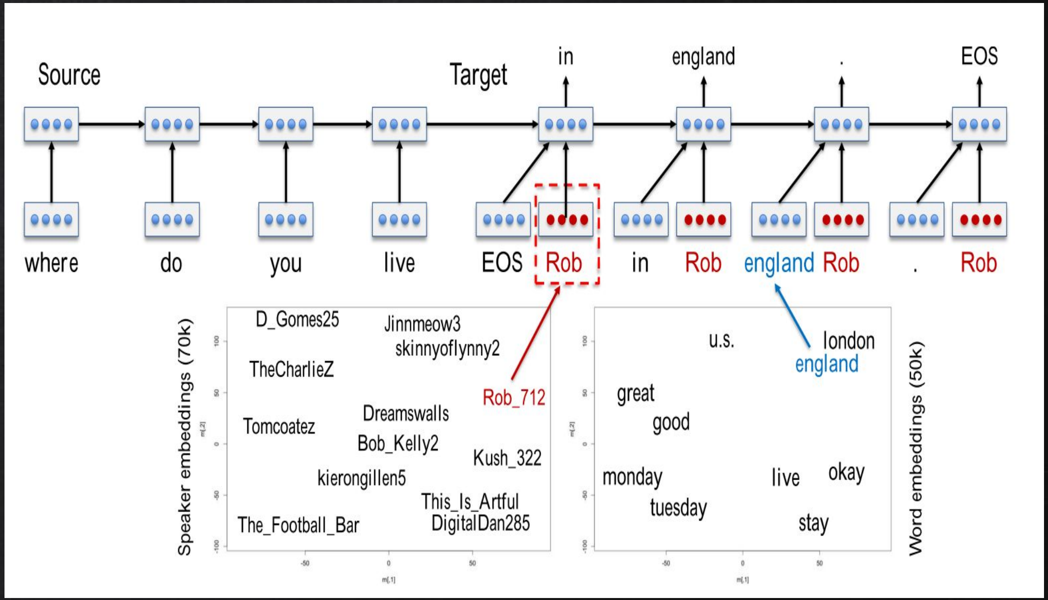


Figure 1: Computational graph for VHRED model. Rounded boxes represent (deterministic) real-valued vectors. Variables **z** represent latent stochastic variables.

"A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues" (2016)

Persona-Based Conversation Model

- Speaker embedding
- 채팅 데이터에 없는 정보를 추론할 수 있음
- Ex) 영국에 사는 Rob / Josh
  - 둘 다 영국 거주
    => 비슷한 speaker embedding
  - Rob 만 챗봇에게 자기가 영국에 산다고 대답한 적 있음
  - 하지만 speaker embedding 이 비슷하기 때문에 Josh도 영국에 살 것이라고 유추할 수 있음



"A Persona-Based Neural Conversation Model" (2016)

CoVe (Context Vector)

- Transfer Learning
  - 대용량 데이터에서 학습한 모델은 다른 모델에게 학습한 내용을 잘 전달할 수 있다
  - Pretraining / Fine-tuning
- MT-LSTM
  - Seq2Seq + Attention 번역 모델은 임의의 텍스트를 좋은 distribution의 벡터를 생성할 것이다
  - 그럼 그냥 그 번역 모델의 Encoder를 Feature Extractor 로 사용하자!

Glove

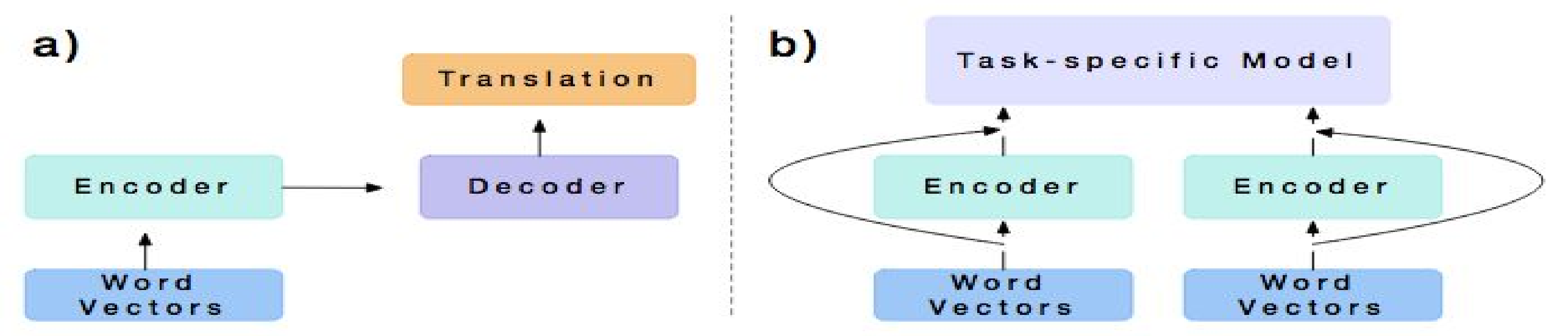


Figure 11: A general overview of how we a) train an encoder and b) reuse it as part of a new model.

Glove

"Learned in Translation: Contextualized Word Vectors" (2017)

CoVe (Context Vector)

Task	Prior State of the Art	Ours
SST-2	**91.8** (Radford et al., 2017)	90.3
SST-5	53.1 (Munkhdalai and Yu, 2016b)	**53.7**
IMDb	**94.1** (Miyato et al., 2017)	91.8
TREC-6	**96.1** (Zhou et al., 2016)	95.8
TREC-50	**91.6** (Van-Tu and Anh-Cuong, 2016)	90.2
SNLI	88.0 (Chen et al., 2016)	**88.1**
SQuAD	82.5 (Wang et al., 2017)	**82.8**

Table 2: Test performance comparison to other machine learning approaches at time of testing (7/12/17).

"Learned in Translation: Contextualized Word Vectors" (2017)

3.

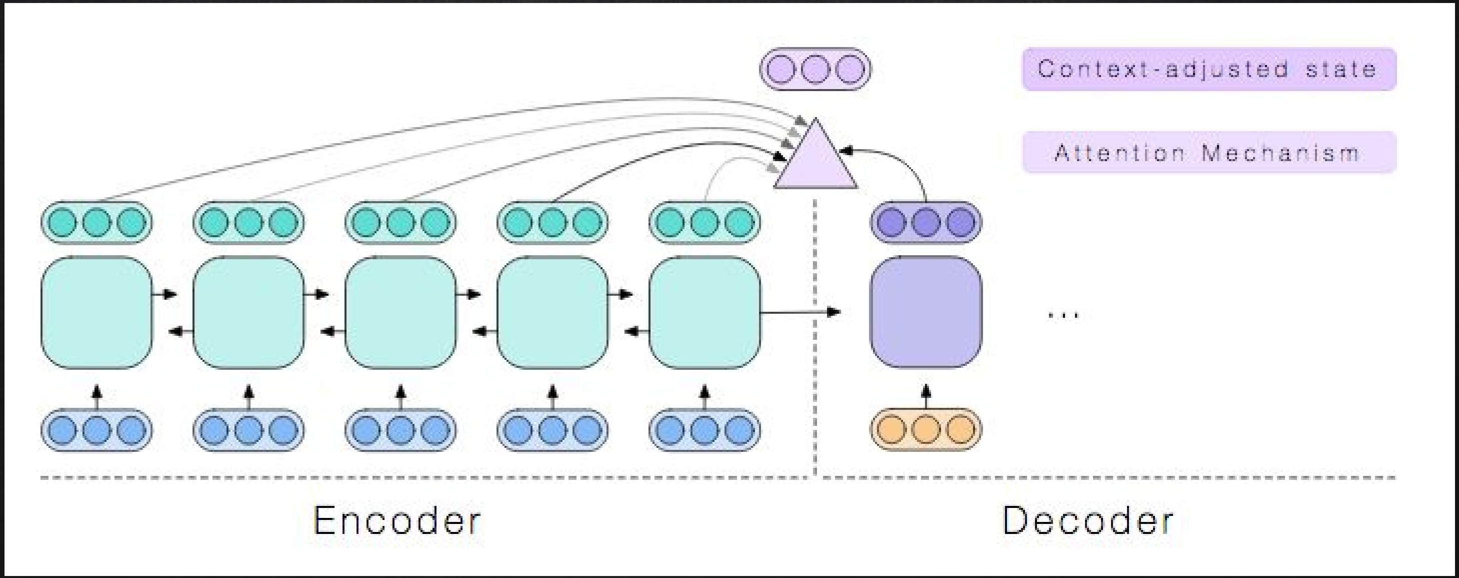
Attention Mechanism

Attention Mechanism
Different attention scoring
Global / Local attention

Attention

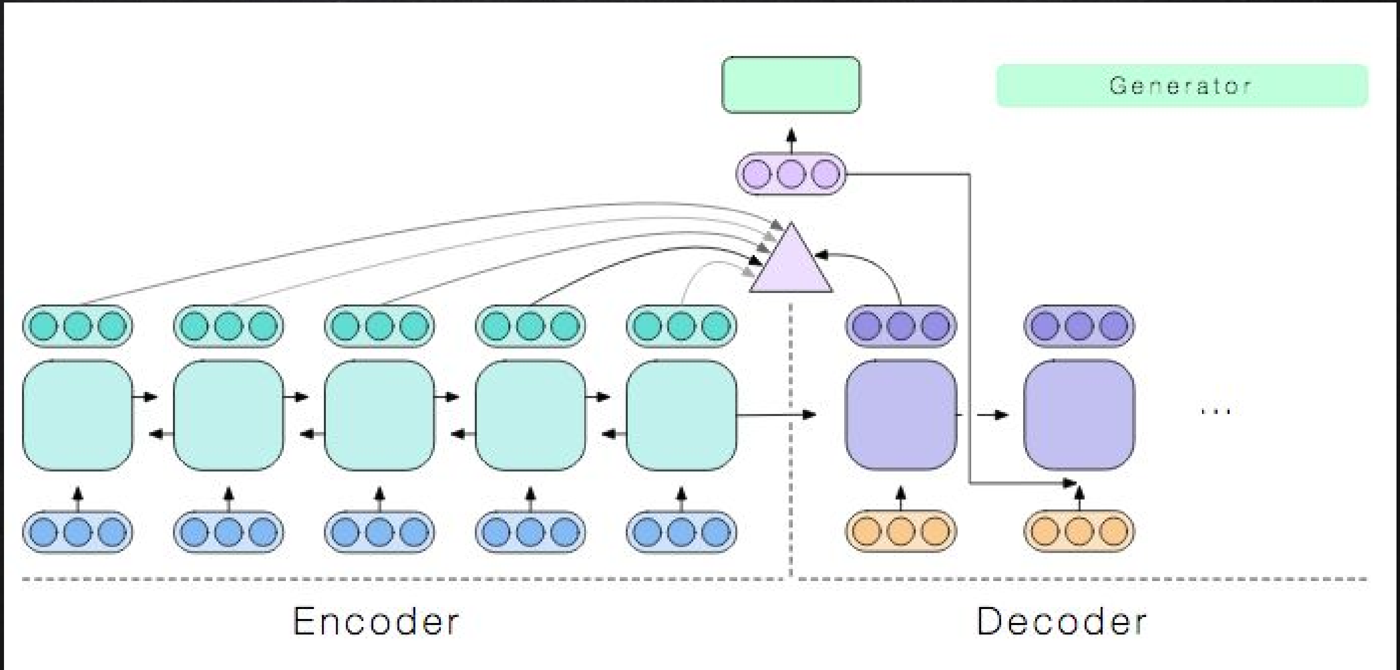
- Encoder RNN 의 마지막 Output 만으로 Source 문장을 나타내는 것은 상당한 정보 손실
- Decoder의 매 스텝마다 Source 문장을 문맥에 맞게 새로 백터화, 이를 바탕으로 단어 생성
- [Interactive demo of distill.pub]()

Attention



"Learned in Translation: Contextualized Word Vectors" (2017)

Attention



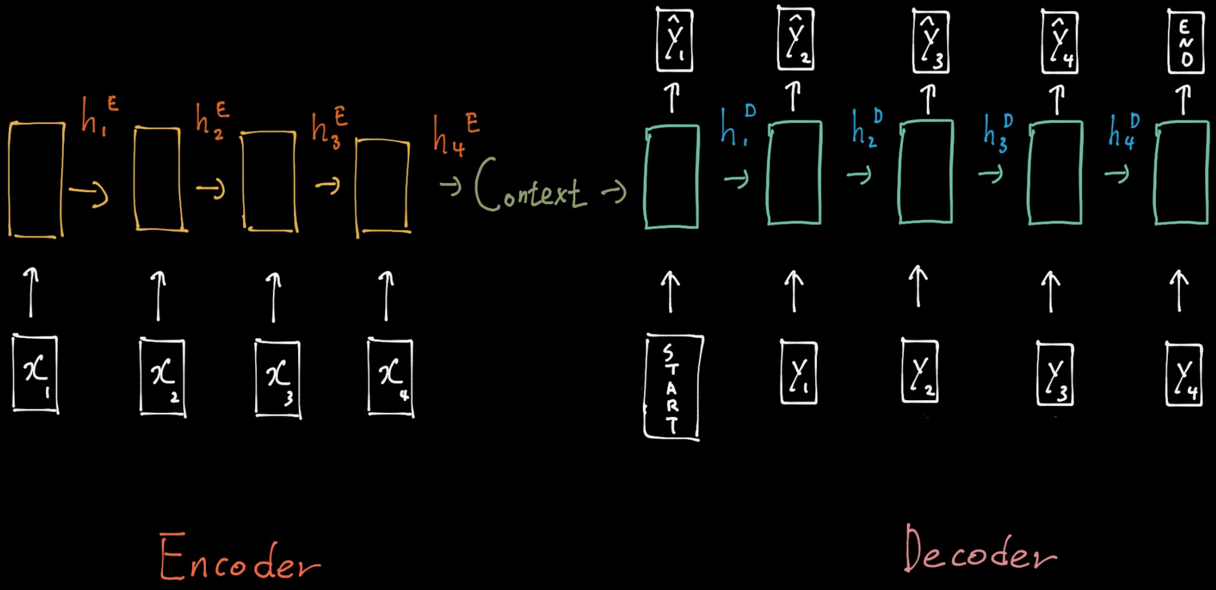
"Learned in Translation: Contextualized Word Vectors" (2017)

RNN Encoder - Decoder (Seq2Seq)

Batch x hidden

Batch x embed

Batch x Vocab



Batch x Vocab

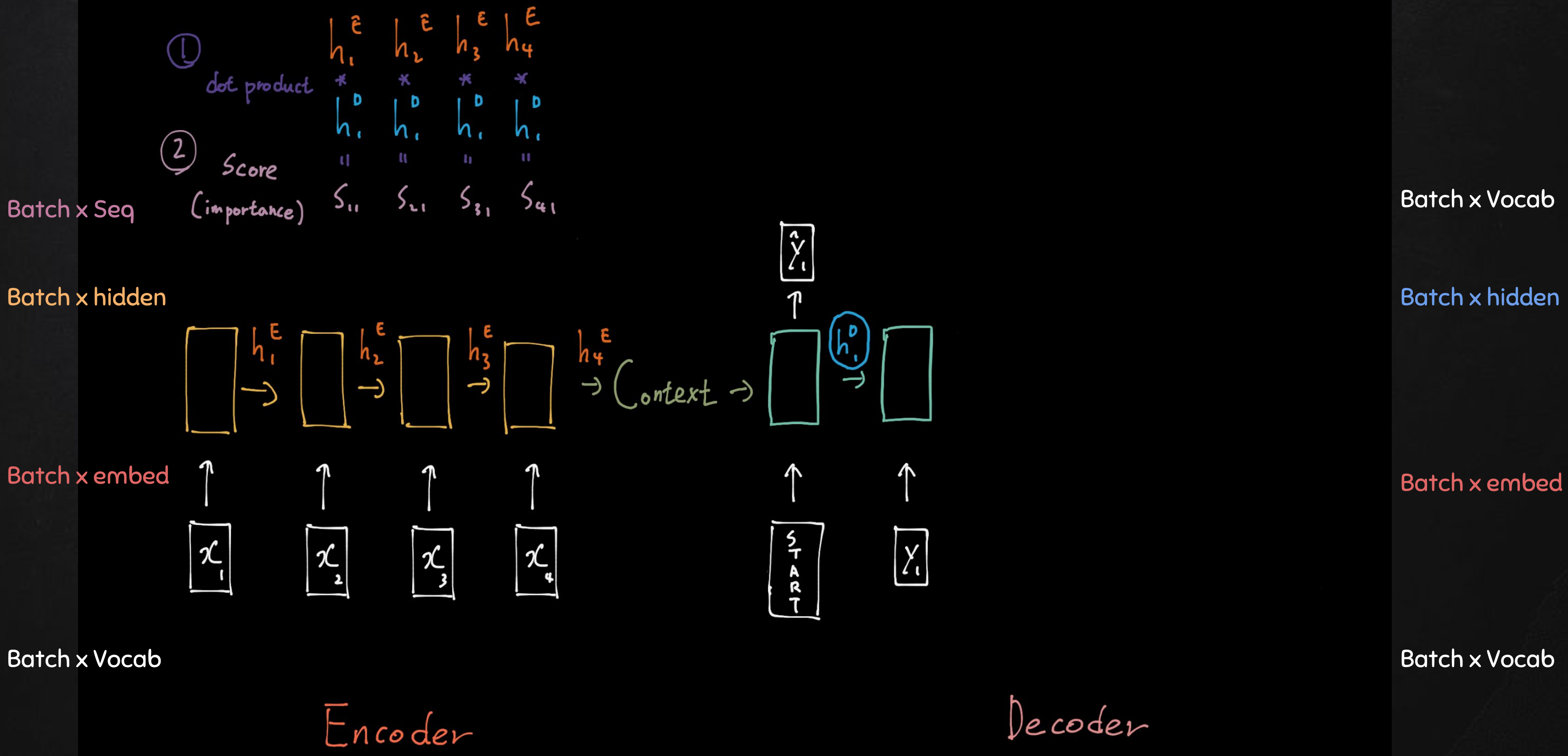
Batch x hidden

Batch x embed

Batch x Vocab

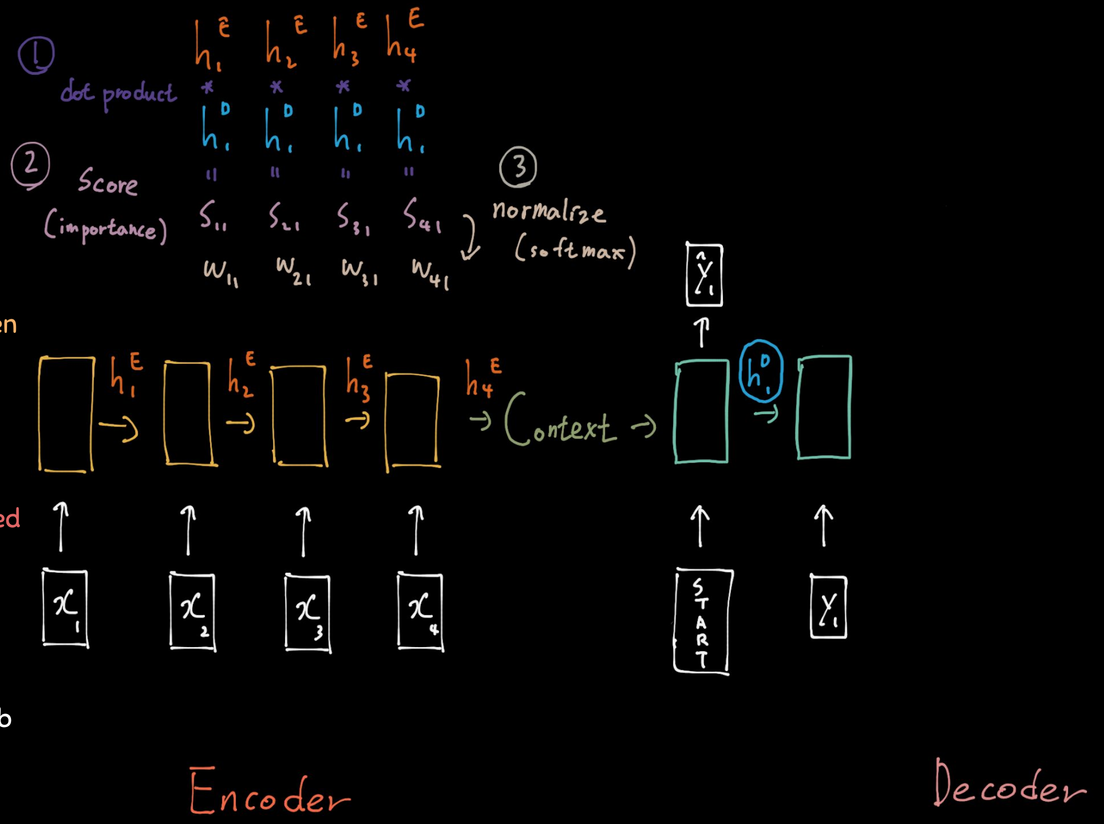
"Sequence to Sequence Learning with Neural Networks" (2014)

Seq2Seq with Attention



"Effective Approaches to Attention-based Neural Machine Translation" (2015)

Seq2Seq with Attention



Batch x Seq

Batch x hidden

Batch x embed

Batch x Vocab

Batch x Vocab

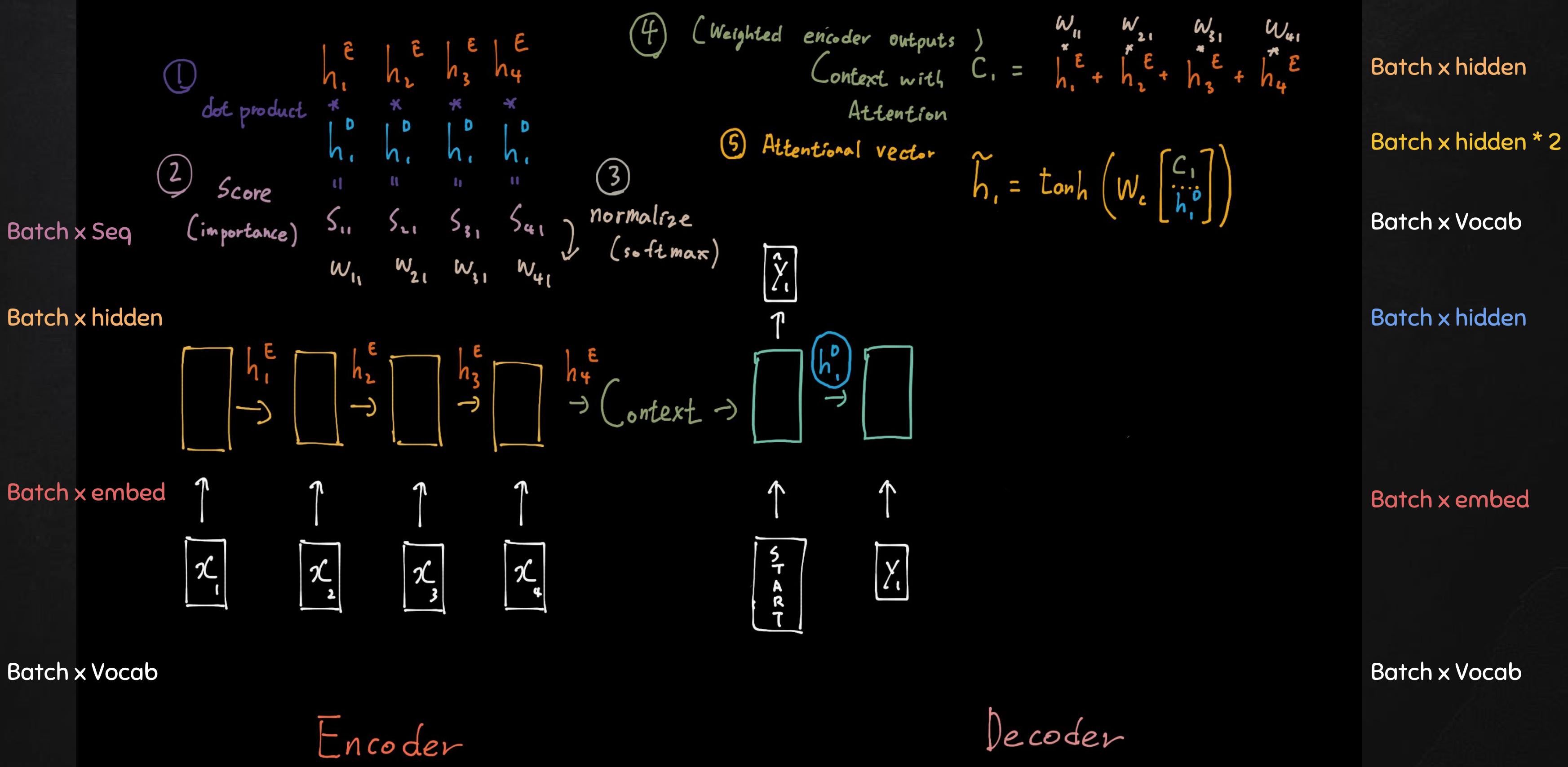
Batch x hidden

Batch x embed

Batch x Vocab

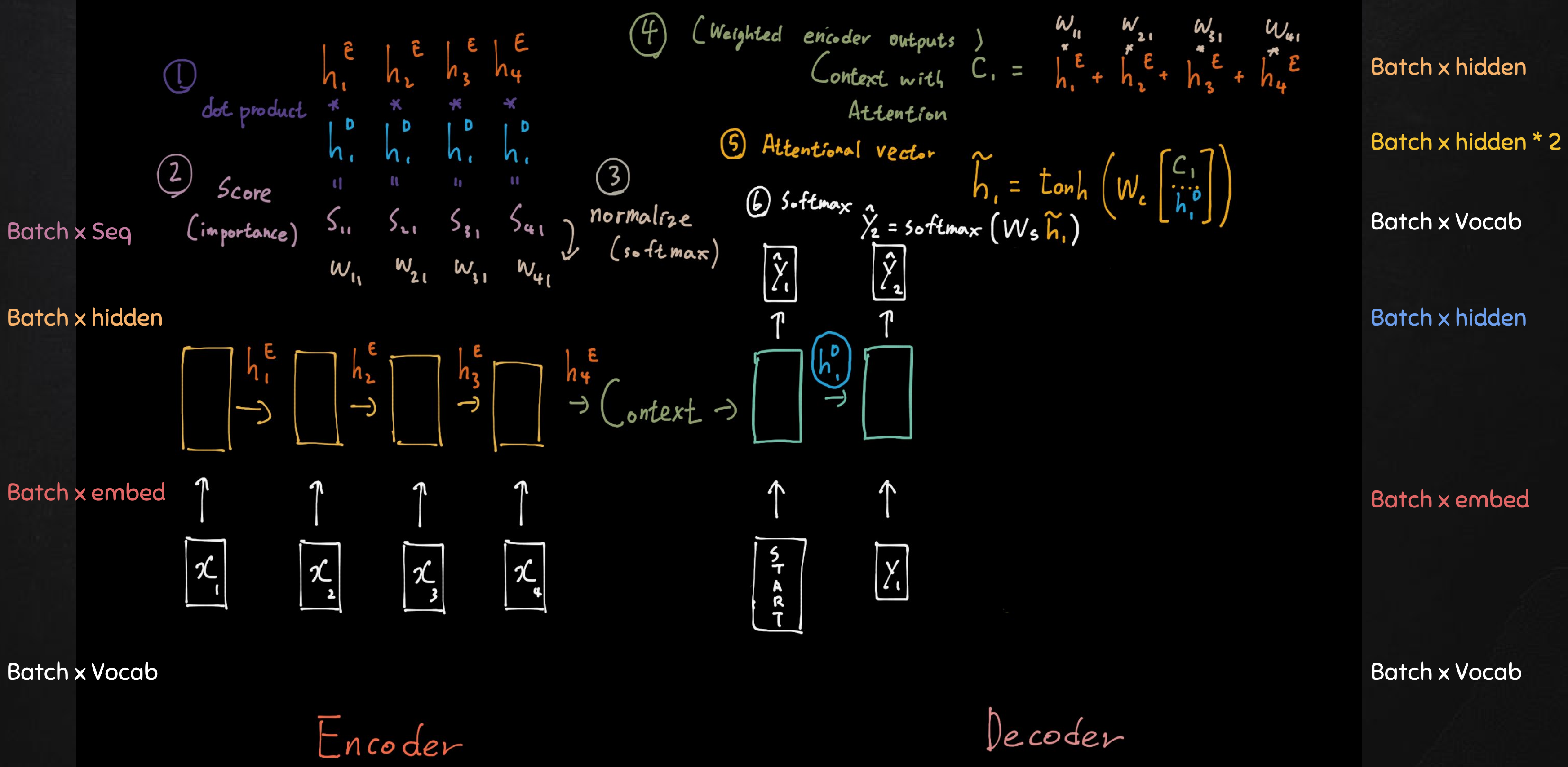
"Effective Approaches to Attention-based Neural Machine Translation" (2015)

Seq2Seq with Attention

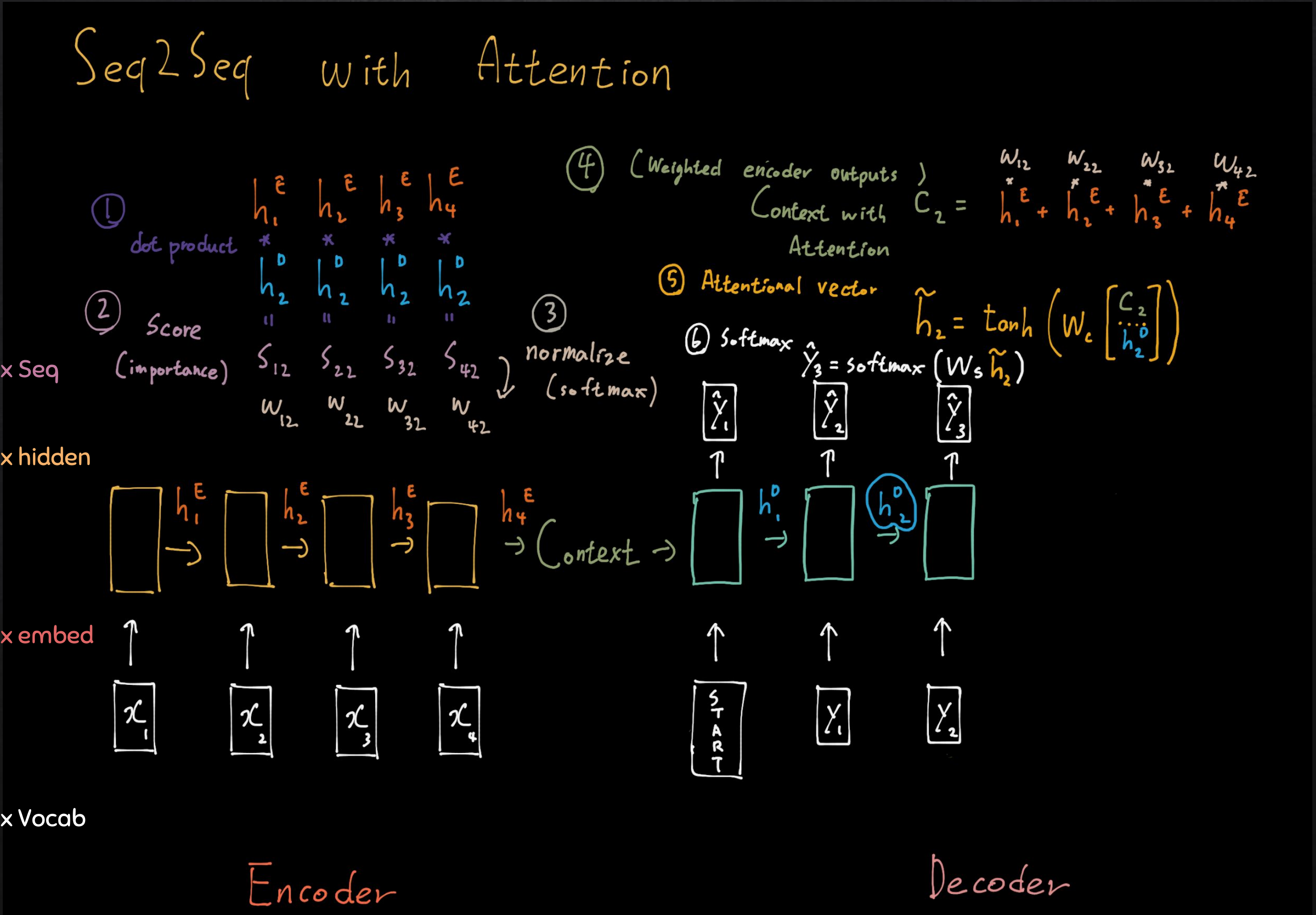


"Effective Approaches to Attention-based Neural Machine Translation" (2015)

Seq2Seq with Attention



"Effective Approaches to Attention-based Neural Machine Translation" (2015)



Batch x Seq

Batch x hidden

Batch x embed

Batch x Vocab

Batch x hidden

Batch x hidden * 2

Batch x Vocab

Batch x hidden

Batch x embed

Batch x Vocab

"Effective Approaches to Attention-based Neural Machine Translation" (2015)

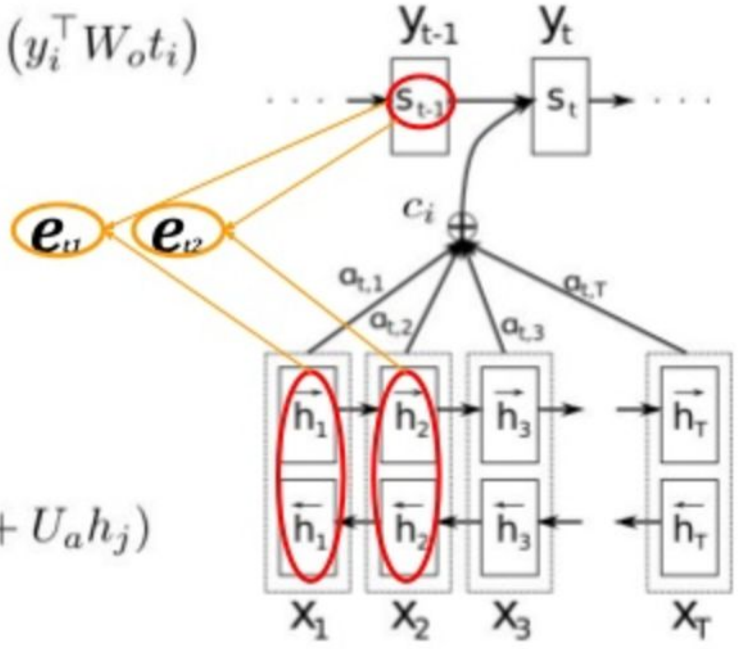
Bahdanau Attention

Softmax $$p(y_i|s_i, y_{i-1}, c_i) \propto \exp\left(y_i^\top W_o t_i\right)$$

Context $$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Weight of h $$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

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



"Neural machine translation by jointly learning to align and translate" (2015)

Different scoring methods

- Luong

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

- h_t: target state
- h_s: source states

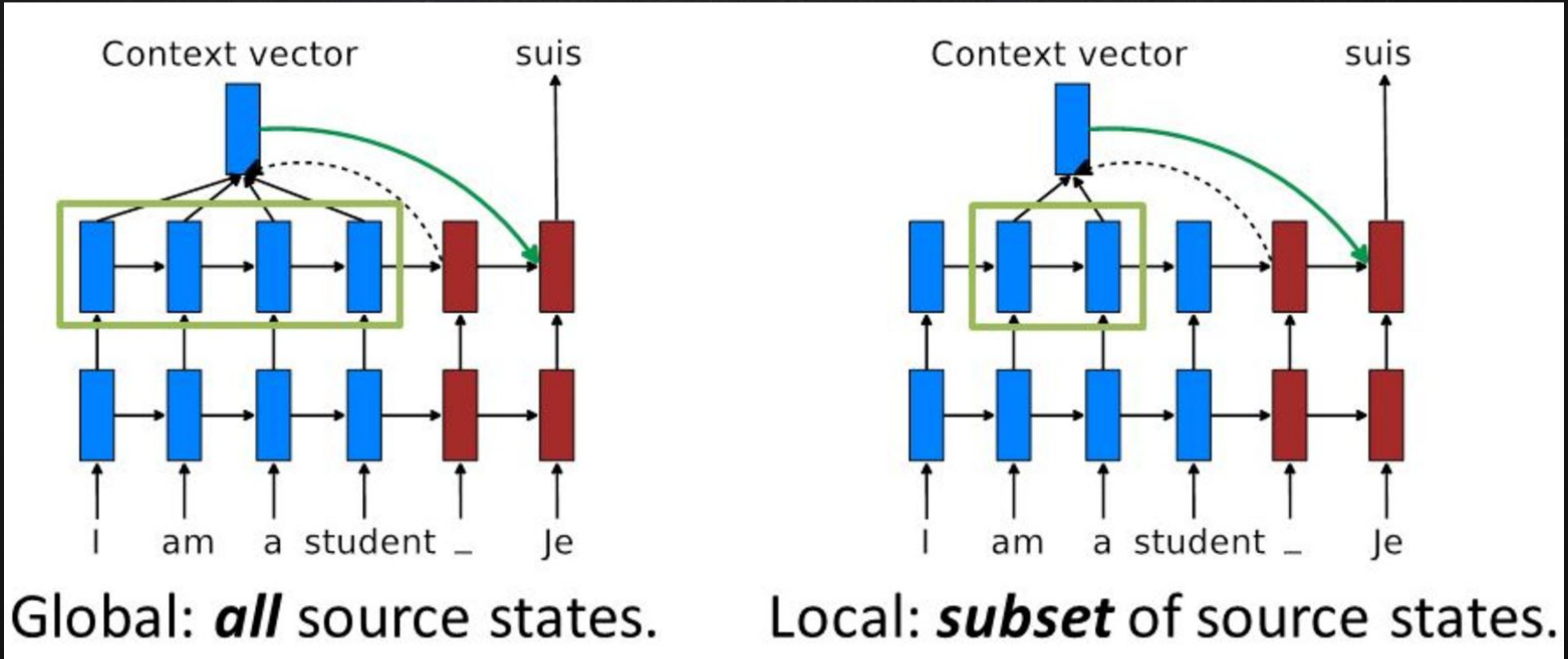
- Bahdanau

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

- s: target state
- h: source state

"Effective Approaches to Attention-based Neural Machine Translation" (2015)

Global and Local Attention



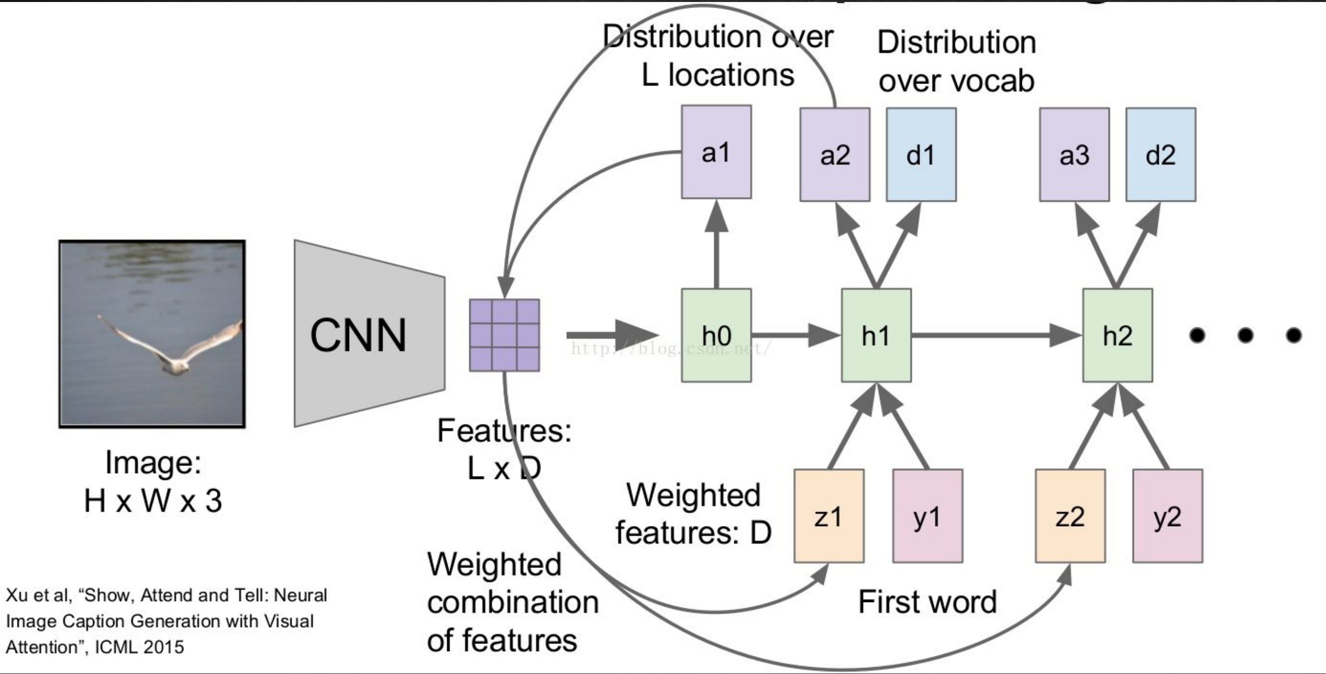
"Effective Approaches to Attention-based Neural Machine Translation" (2015)

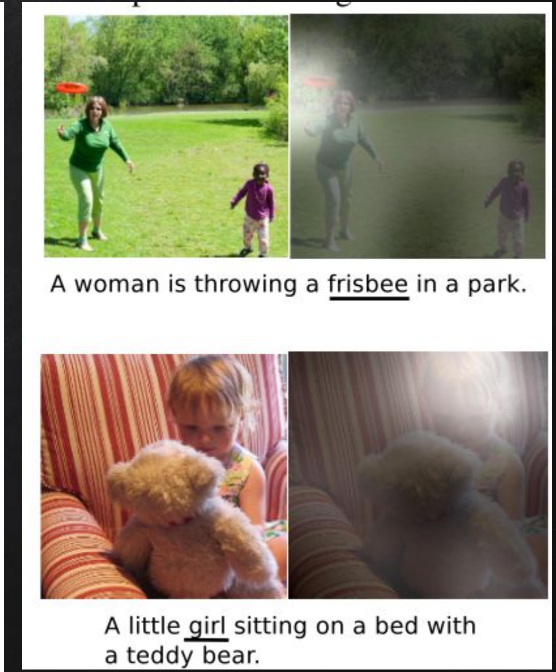
4.

Advanced Attention Mechanism

Image Captioning: Show, Attend and Tell
Pointing Attention: Pointer Networks
Copying Mechanism: CopyNet
Bidirectional Attention: Bidirectional Attention Flow (BIDAF)
Self-Attention: Transformer

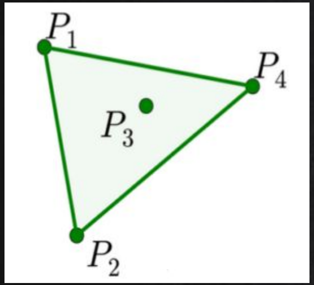
Show, Attend And Tell



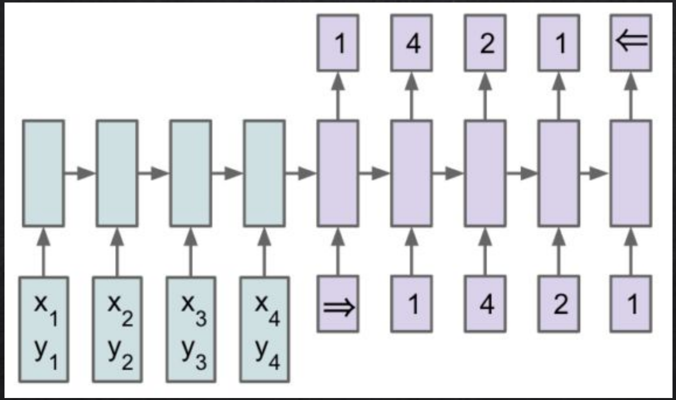


"Show, Attend and Tell: Neural Image Caption Generation with Visual Attention" (2015)

Pointing Attention (Pointer Networks)

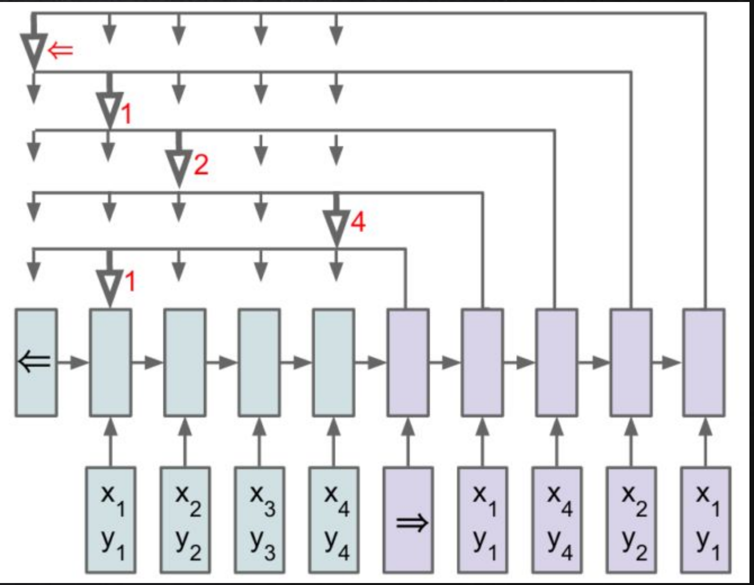


Seq2Seq + Attention



$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \\
a_j^i &= \mathrm{softmax}(u_j^i) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

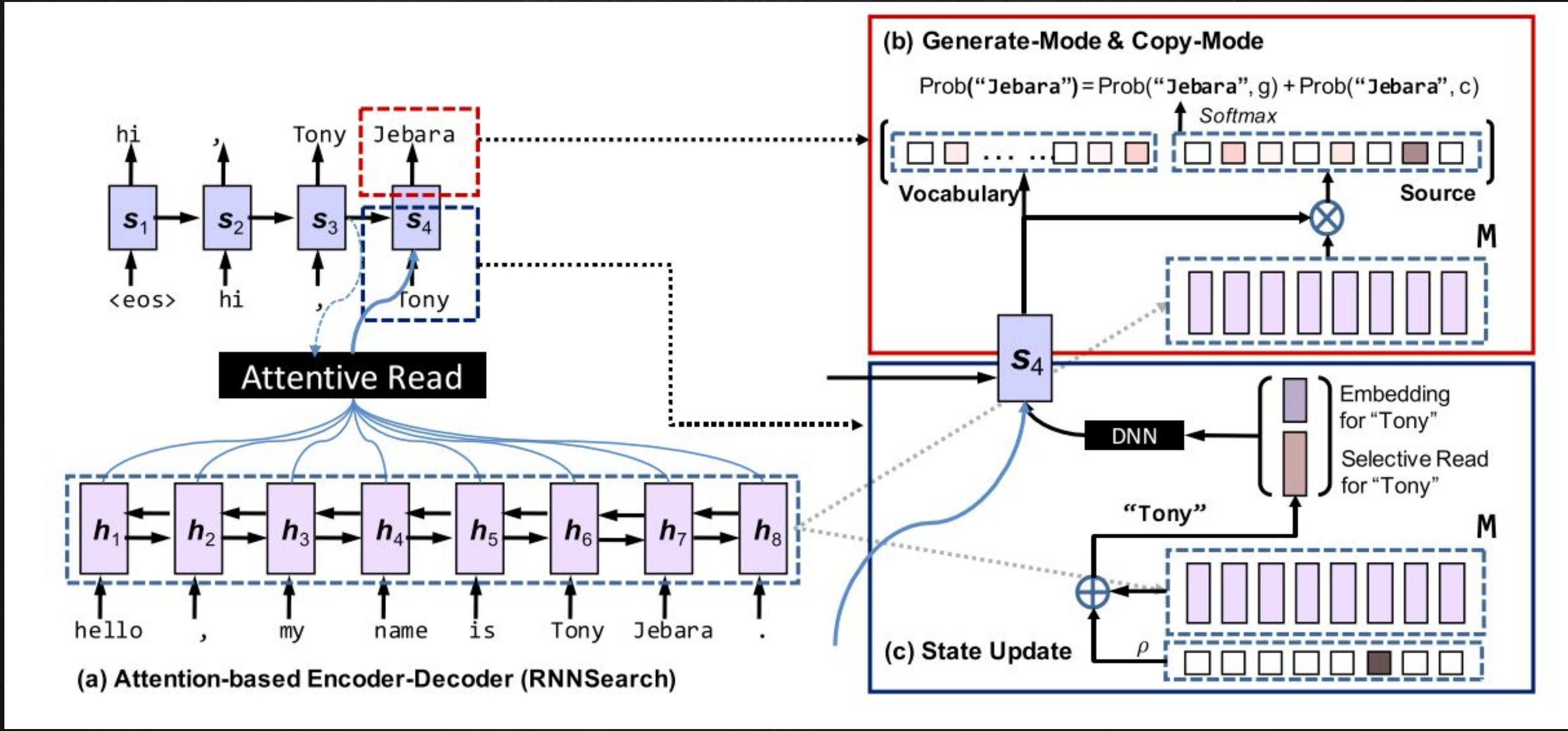
Pointer Networks



$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \\
p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) &= \mathrm{softmax}(u^i)
\end{aligned}
$$

"Pointer Networks" (2016)

Copying Mechanism (CopyNet)



"Incorporating Copying Mechanism in Sequence-to-Sequence Learning" (2016)

Bi-directional Attention Flow (BIDAF)

- Question Answering
  - 2016년 말 SQUAD SOTA
  - 정답은 지문 속에 있는 sub-phrase
  - '시작' 과 '끝' 부분 찾아서 '밑줄 치기'
- Bi-attention
  - 현재 읽는 지문의 단어들과 가장 가까운 문제의 단어는?
  - 현재 읽는 문제의 단어들과 가장 가까운 지문의 단어는?
- Embedding
  - Word-embedding
  - Char-embedding
  - Highway Network
- Decoder
  - Pointer-like Network

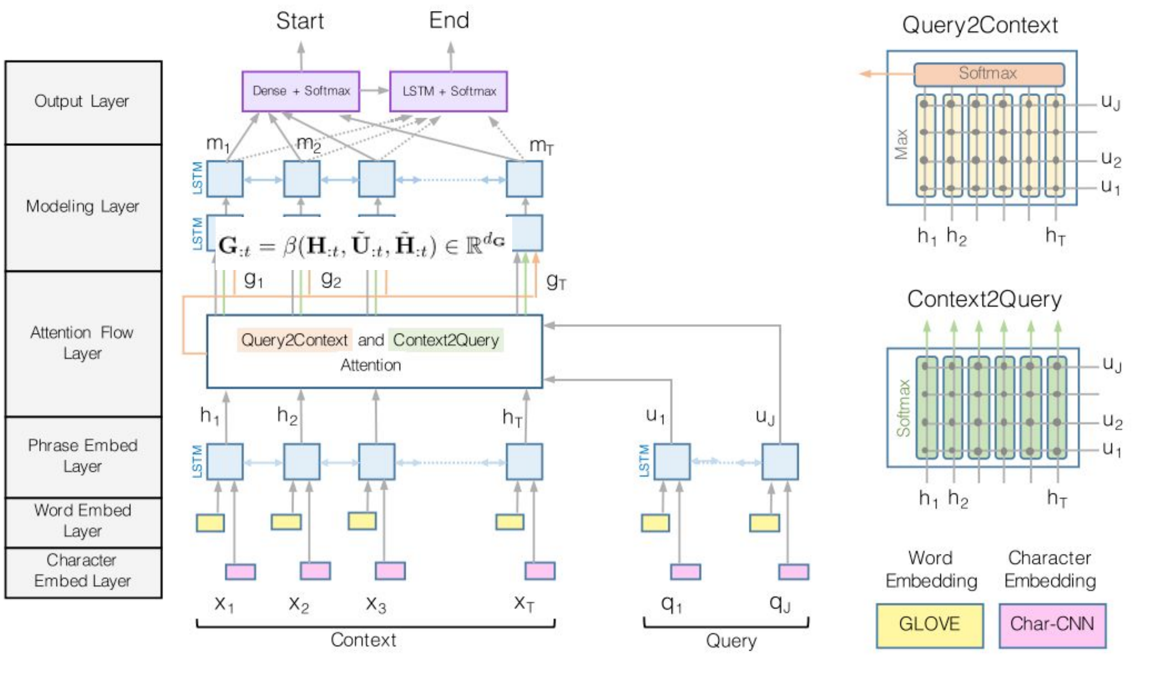


Figure 1: BiDirectional Attention Flow Model *(best viewed in color)*

"Bi-Directional Attention Flow for Machine Comprehension" (2016)

Bi-directional Attention Flow (BIDAF)

	Single Model		Ensemble	
	EM	F1	EM	F1
Logistic Regression Baseline[1]	40.4	51.0	-	-
Dynamic Chunk Reader[2]	62.5	71.0	-	-
Fine-Grained Gating[3]	62.5	73.3	-	-
Match LSTM[4]	64.7	73.7	67.9	77.0
Multi-Perspective Matching[5]	65.5	75.1	68.2	77.2
Dynamic Coattention Networks[6]	66.2	75.9	71.6	80.4
R-Net[7]	**68.4**	**77.5**	72.1	79.7
BiDAF (Ours)	68.0	77.3	**73.3**	**81.1**

(a) Results on the SQuAD test set

	EM	F1
No char embedding	65.0	75.4
No word embedding	55.5	66.8
No C2Q attention	57.7	69.0
No Q2C attention	63.6	73.7
Dynamic attention	63.5	73.6
BiDAF (single)	68.0	77.3
BiDAF (ensemble)	73.3	81.1

(b) Ablations on the SQuAD dev set

	CNN		DailyMail	
	val	test	val	test
Attentive Reader (Hermann et al., 2015)	61.6	63.0	70.5	69.0
MemNN (Hill et al., 2016)	63.4	6.8	-	-
AS Reader (Kadlec et al., 2016)	68.6	69.5	75.0	73.9
Stanford AR (Chen et al., 2016)	68.6	69.5	75.0	73.9
DER Network (Kobayashi et al., 2016)	71.3	72.9	-	-
Iterative Attention (Sordoni et al., 2016)	72.6	73.3	-	-
EpiReader (Trischler et al., 2016)	73.4	74.0	-	-
GAReader (Dhingra et al., 2016)	73.0	73.8	76.7	75.7
AoA Reader (Cui et al., 2016)	73.1	74.4	-	-
ReasoNet (Shen et al., 2016)	72.9	74.7	77.6	76.6
BiDAF (Ours)	**76.3**	**76.9**	**80.3**	**79.6**
MemNN* (Hill et al., 2016)	66.2	69.4	-	-
ASReader* (Kadlec et al., 2016)	73.9	75.4	78.7	77.7
Iterative Attention* (Sordoni et al., 2016)	74.5	75.7	-	-
GA Reader* (Dhingra et al., 2016)	76.4	77.4	79.1	78.1

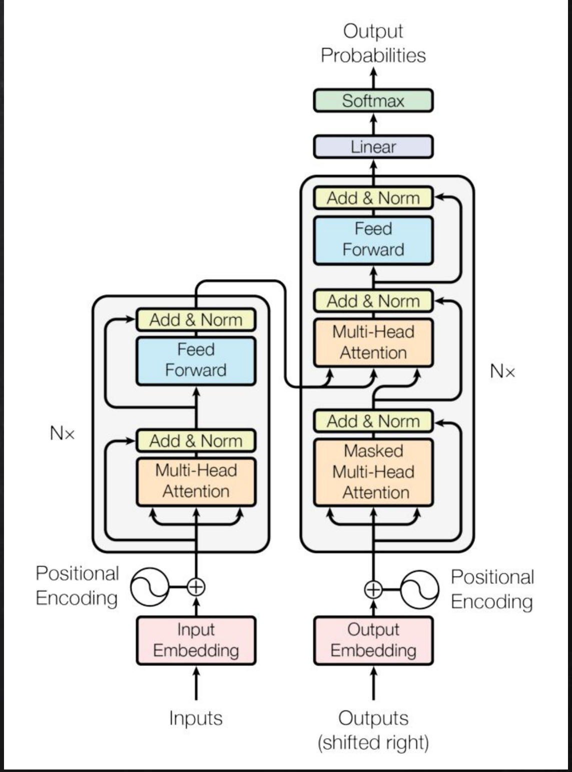
"Bi-Directional Attention Flow for Machine Comprehension" (2016)

Transformer

- ✘ Machine Translation
- ✘ Self-attention Encoder-Decoder
- ✘ Multi-layer "Scaled Dot-product Multi-head" attention
- ✘ Positional Embeddings
- ✘ Residual Connection
- ✘ Byte-pair Encoding (BPE)

Translation Model	Training time	BLEU (difference from baseline)
Transformer (T2T)	3 days on 8 GPU	**28.4 (+7.8)**
SliceNet (T2T)	6 days on 32 GPUs	26.1 (+5.5)
GNMT + Mixture of Experts	1 day on 64 GPUs	26.0 (+5.4)
ConvS2S	18 days on 1 GPU	25.1 (+4.5)
GNMT	1 day on 96 GPUs	24.6 (+4.0)
ByteNet	8 days on 32 GPUs	23.8 (+3.2)
MOSES (phrase-based baseline)	N/A	20.6 (+0.0)

BLEU scores (higher is better) on the standard WMT English-German translation task.

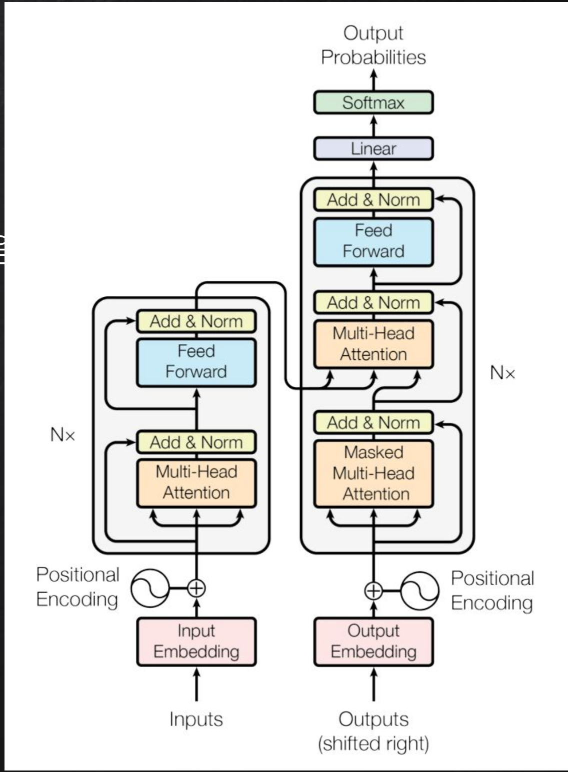


"Attention is All You need" (2017)

Transformer

- 기존의 CNN/RNN 기반 Language Understanding의 단점
  - Path-length 가 멀다
    - 네트워크 내에서 한 단어와 다른 단어 node 사이의 거리
    - 멀수록 Long-Term dependency 잡아내기 힘듦
  - Dilated Convolution, Attention 등으로 path length를 줄여 왔

- Transformer
  - self-attention만으로 이루어진 encoder-decoder
  - 주어진 token 개수만큼 hidden representation 생성



"Attention is All You need" (2017)

Transformer

- ✘ Positional Encoding
  - ○ 토큰의 순서를 모델링
    - ■ i: 벡터의 몇 번째 원소인지

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

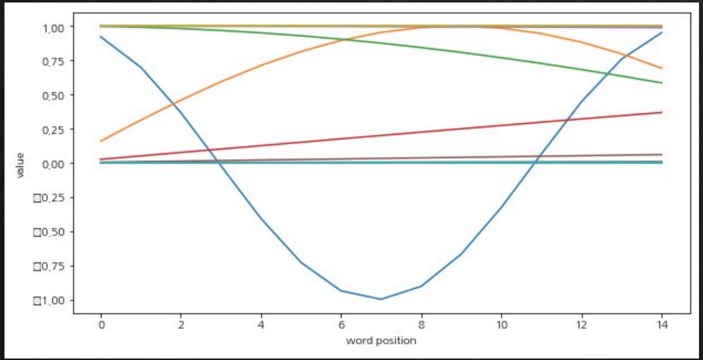
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- ✘ [OpenNMT-py's implementation]
- ✘ [Visualization]

```python
class PositionalEncoding(nn.Module):

    def __init__(self, dropout, dim, max_len=5000):
        pe = torch.arange(0, max_len).unsqueeze(1).expand(max_len, dim)
        div_term = 1 / torch.pow(10000, torch.arange(0, dim * 2, 2) / dim)
        pe = pe * div_term.expand_as(pe)
        pe[:, 0::2] = torch.sin(pe[:, 0::2])
        pe[:, 1::2] = torch.cos(pe[:, 1::2])
        pe = pe.unsqueeze(1)
        super(PositionalEncoding, self).__init__()
        self.register_buffer('pe', pe)
        self.dropout = nn.Dropout(p=dropout)

    def forward(self, emb):
        # We must wrap the self.pe in Variable to compute, not the other
        # way - unwrap emb(i.e. emb.data). Otherwise the computation
        # wouldn't be watched to build the compute graph.
        emb = emb + Variable(self.pe[:emb.size(0), :1, :emb.size(2)]
                             .expand_as(emb), requires_grad=False)
        emb = self.dropout(emb)
        return emb
```



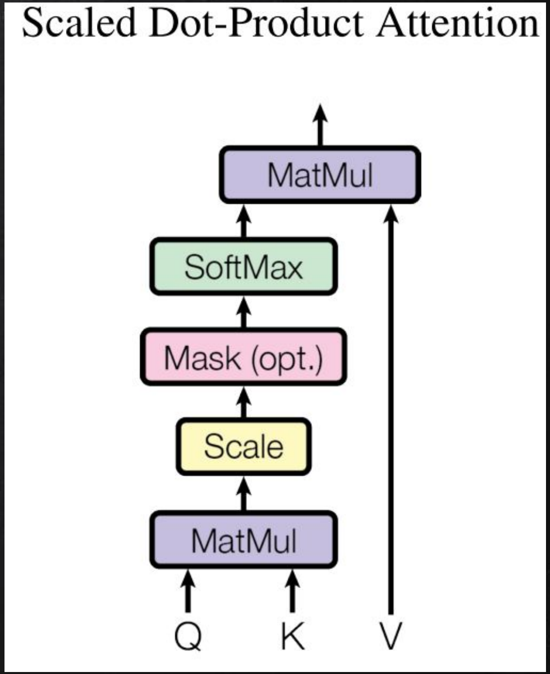
"Attention is All You need" (2017)

Transformer

- Attention
  - Attention layer를 encoder/decoder에 6겹 쌓음
  - 3개의 입력
    - Q, K, V (Query, Key, Value)
    - End-to-End Memory Networks 와 유사

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

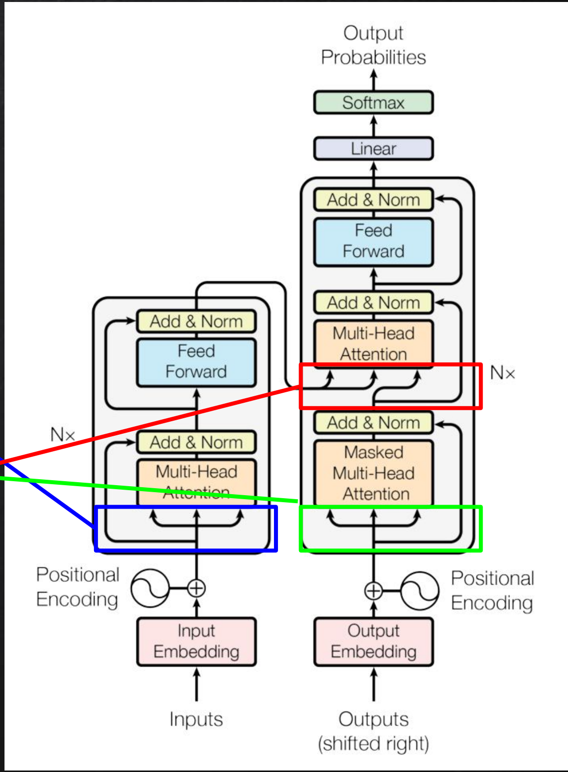
  - Attention Weight
    - Q, K의 dot product & softmax
    - $d_k^{0.5}$로 scaling (smoothing)
      - 자기 자신에만 attention 쏠리는 것 방지
  - V 벡터들의 weighted sum 출력



"Attention is All You need" (2017)

Transformer

- Attention 입력 문장
  - Encoder
    - Q, K, V 모두 source sentence
  - Decoder
    - 첫 레이어
      - Q, K, V 모두 target sentence
    - 이후 레이어
      - Q: target sentence
      - K, V: source sentence



"Attention is All You need" (2017)

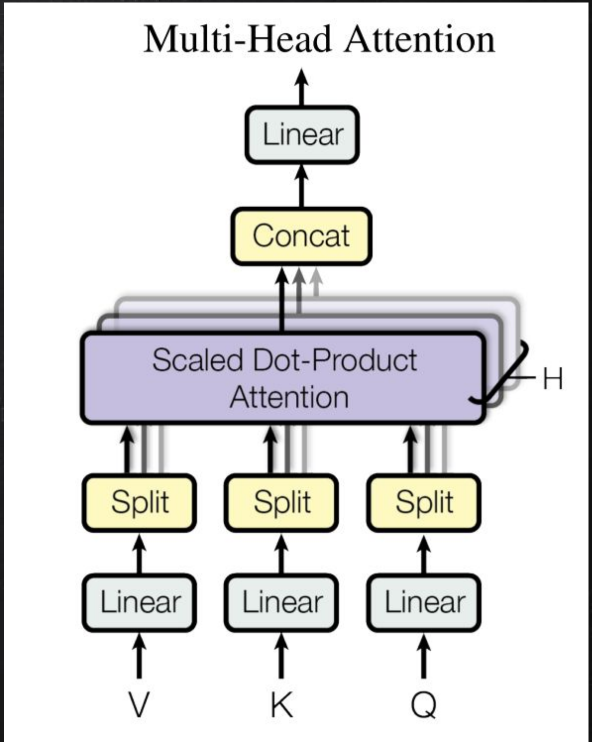
Transformer

- Multi-head Attention
  - 이전 레이어의 출력
    - Q, K, V
    - 모두 d_{model} 차원
  - h개의 W^Q_i, W^K_i, W^V_i 를 이용해서 d_k, d_k, d_v 차원으로 projection

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_h)$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

  - i = 1..h
    - $Q_i = Q$ @ W^Q_i (d_{model} => d_k)
    - $K_i = K$ @ W^K_i (d_{model} => d_k)
    - $V_i = V$ @ W^V_i (d_{model} => d_v)
  - h개의 attention 결과를 concatenation (Inception과 비슷)
- 논문에서는 아래의 값 사용
  - h=8
  - $d_k = d_v = d_{model} / h = 64$

"Attention is All You need" (2017)



Transformer

- 기타
  - Point-wise Feed-Forward
    - 2-layer NN + ReLU
  - Layer norm / Residual Connection
    - Sublayer(x) = FFN(Multi-head Attention(x)
  - Embedding
    - d_{model} * 0.5
  - Optimizer
    - Adam
  - Regularization
    - Residual Dropout
      - p = 0.1
    - Attention Dropout
    - Label smoothing
      - e = 0.1

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

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

$$\mathrm{Attention}(Q, K, V) = \mathrm{dropout}(\mathrm{softmax}(\frac{QK^T}{\sqrt{d}}))V$$

"Attention is All You need" (2017)

Transformer

- Complexity / Maximum Path Length
  - n: sequence length
  - r: restricted size of the neighborhood (local attention)

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

- BLEU Score on WMT 2014

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [16]	23.75			
Deep-Att + PosUnk [35]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [34]	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 [29]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [35]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [34]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

"Attention is All You need" (2017)

Seq2Seq Implementations in PyTorch

- https://github.com/spro/practical-pytorch/tree/master/seq2seq-translation
- https://github.com/OpenNMT/OpenNMT-py
- https://github.com/eladhoffer/seq2seq.pytorch
- https://github.com/IBM/pytorch-seq2seq
- https://github.com/allenai/allennlp

- Also, check out the curated 'Awesome' lists.
  - https://github.com/ritchieng/the-incredible-pytorch
  - https://github.com/bharathgs/Awesome-pytorch-list

THANKS!

Any questions?