텐서플로와 머신러닝으로 시작하는 자연어처리

Lecture 6
2019. 07. 06

Attention?!

Attention?!

- What is Attention Mechanism
- Why do we need Attention
- Papers
  - Neural Machine Translation by Jointly Learning to Align and Translate
- Various Attention Mechanism
- Another methods to enrich NMT performance

What is Attention

What is Attention Mechanism

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

ex) 감정 분석 Task

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

ex) 감정 분석 Task

Input: 이 영화는 감독이 진짜 잘 만든 영화인 것 같다.

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

ex) 감정 분석 Task

Input: 이 영화는 감독이 진짜 잘 만든 영화인 것 같다.

Output: ?

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

ex) 감정 분석 Task

Input: 이 영화는 감독이 진짜 잘 만든 영화인 것 같다.

Output: ?

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

ex) 감정 분석 Task

Input: 이 영화는 감독이 진짜 잘 만든 영화인 것 같다.

Output: ?

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

ex) 감정 분석 Task

Input: 이 영화는 감독이 진짜 잘 만든 영화인 것 같다.

Output: 긍정

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

Image Classification

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

주어진 목적에 맞는 Source에 집중

→ 개? 고양이?

Image Classification

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

Image Classification

개, 고양이 구분

주어진 목적에 맞는 Source에 집중

개? 고양이?

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동

개, 고양이 구분 각 픽셀값

주어진 목적에 맞는 Source에 집중

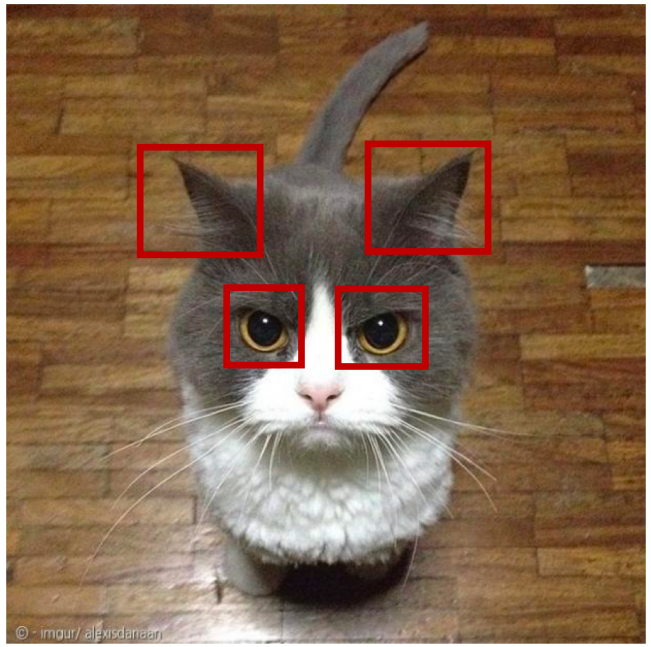
개? 고양이?

Image Classification

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동



주어진 목적에 맞는 Source에 집중

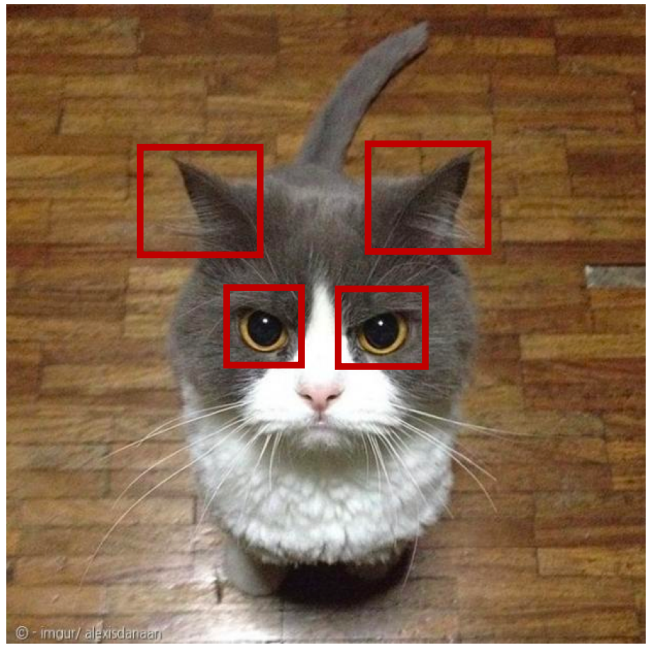
→ 개? 고양이?

Image Classification

What is Attention Mechanism

attention 미국·영국 [əˈtenʃn] 영국식 ★★

1. 주의 (집중), 주목
2. 관심, 흥미
3. (관심을 끌기 위한) 행동



주어진 목적에 맞는 Source에 집중

→ 개? 고양이!

Image Classification

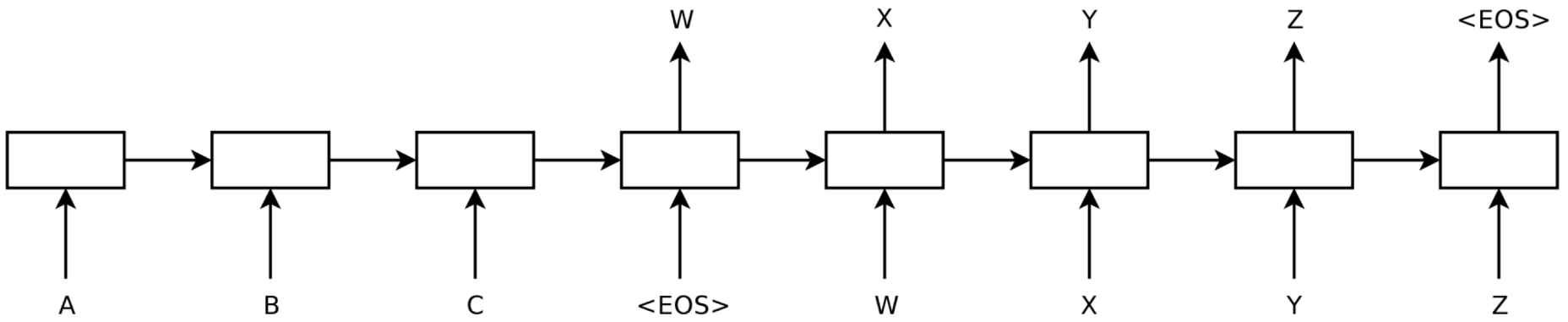
Why do we need Attention

Why do we need Attention

Limitation of Sequence to Sequence Model

Why do we need Attention

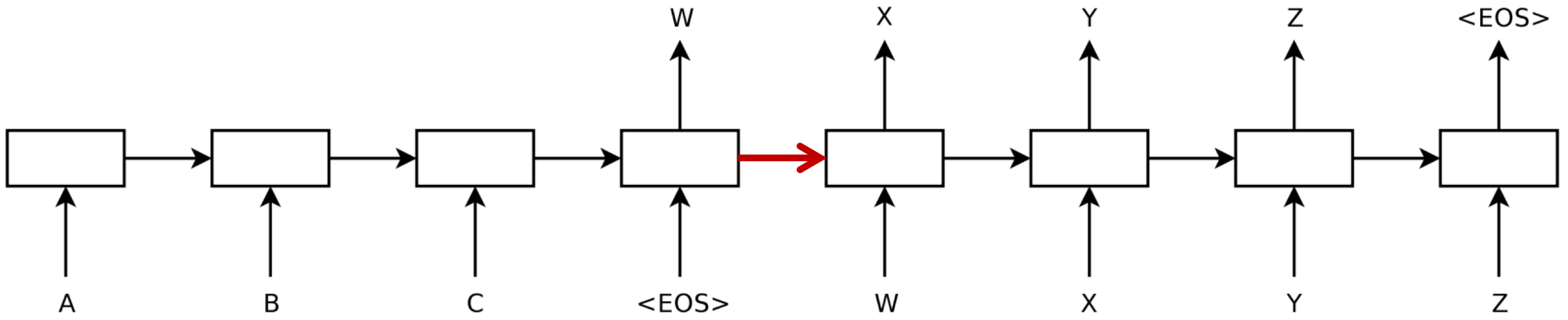
Limitation of Sequence to Sequence Model



Why do we need Attention

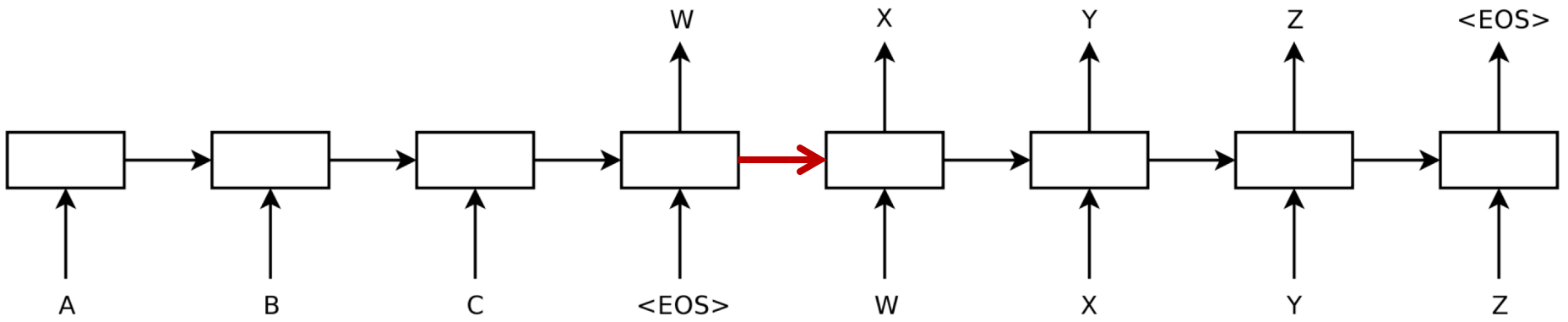
Limitation of Sequence to Sequence Model

인코더 부분 화살표 수정



Why do we need Attention

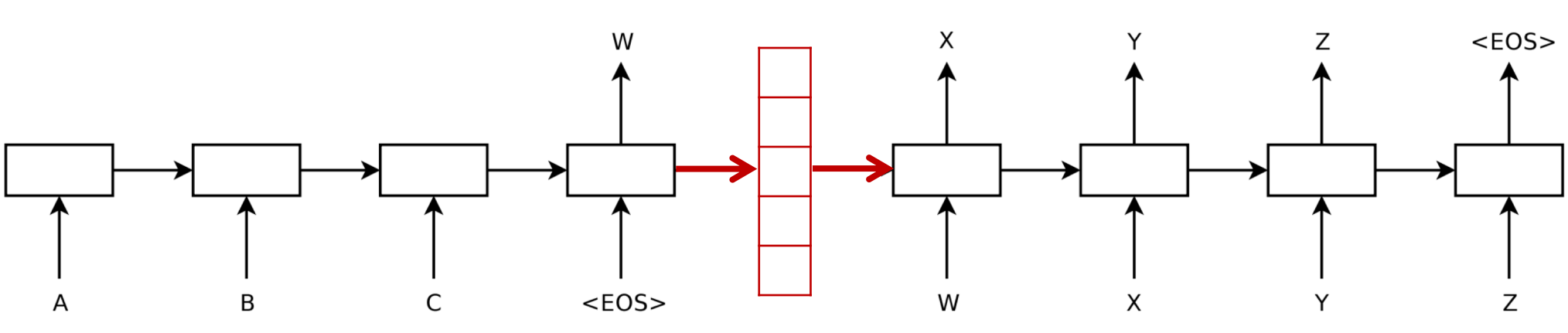
Limitation of Sequence to Sequence Model



모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

Limitation of Sequence to Sequence Model



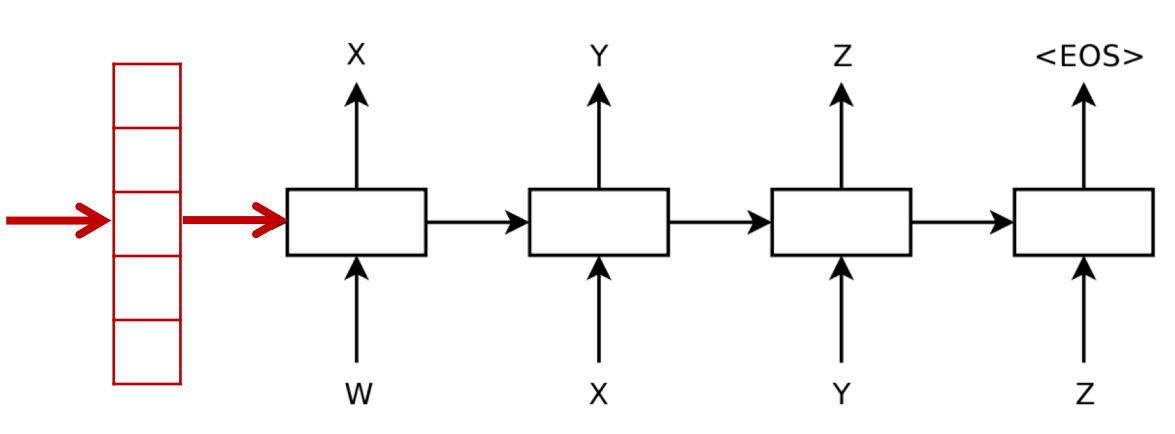
모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

Limitation of Sequence to Sequence Model

Input

The Sequence to Sequence model is good



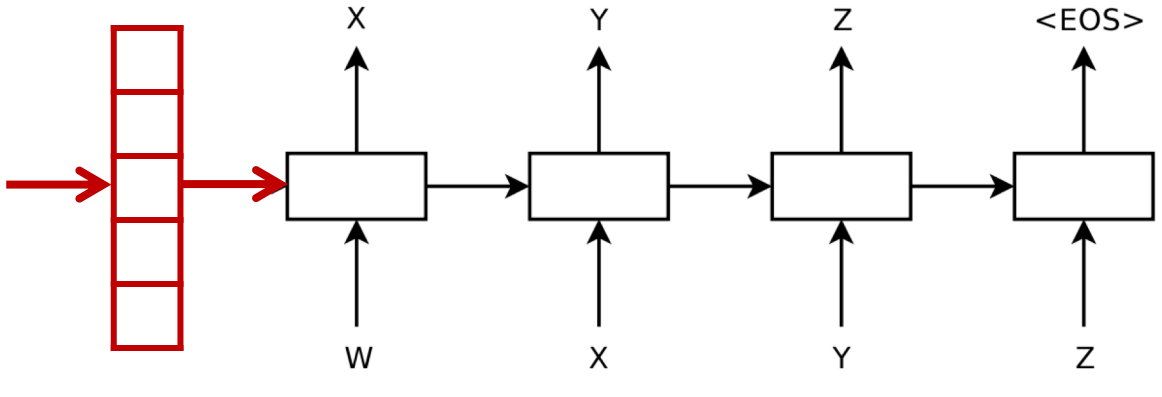
모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

Limitation of Sequence to Sequence Model

Input

The Sequence to Sequence model is good



모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

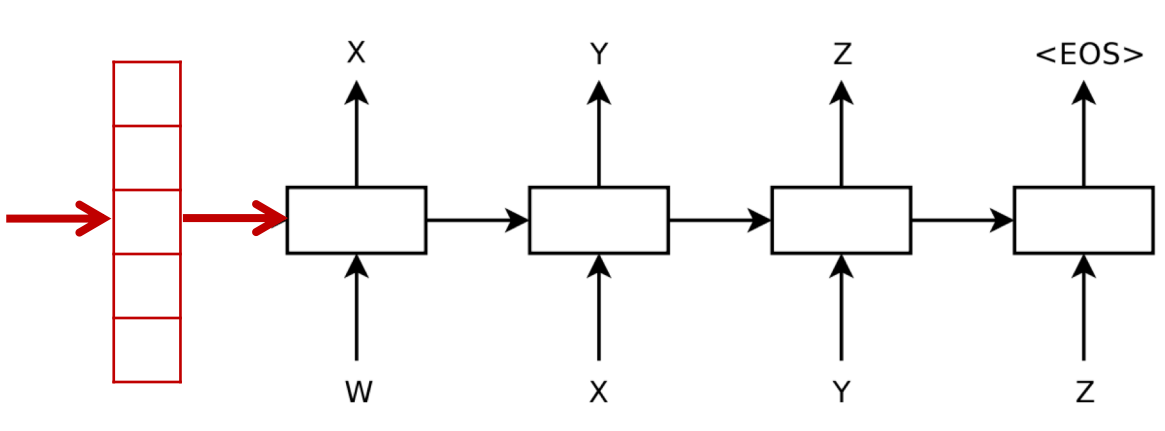
Limitation of Sequence to Sequence Model

Input

The Sequence to Sequence which is

made by Cho et al. at NYU University

model is good for Machine Translation



모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

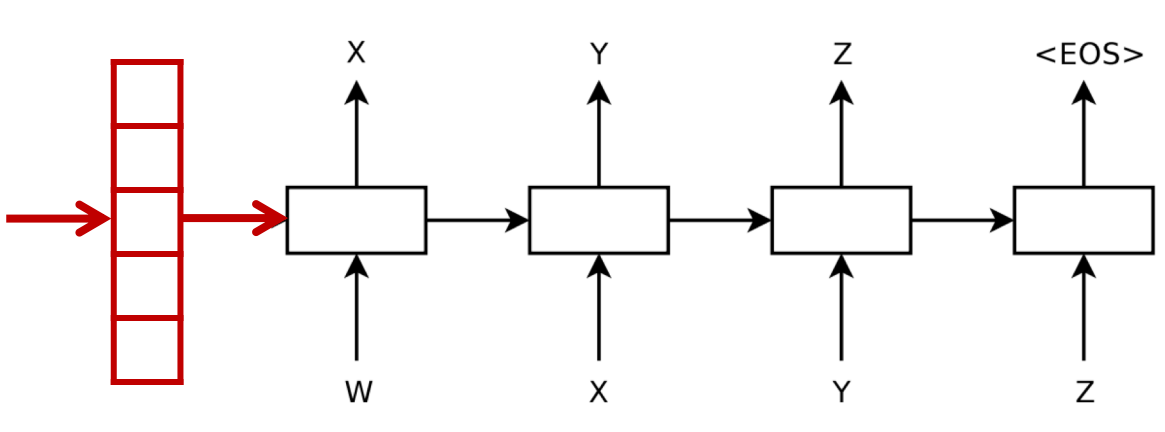
Limitation of Sequence to Sequence Model

Input

The Sequence to Sequence which is

made by Cho et al. at NYU University

model is good for Machine Translation



모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

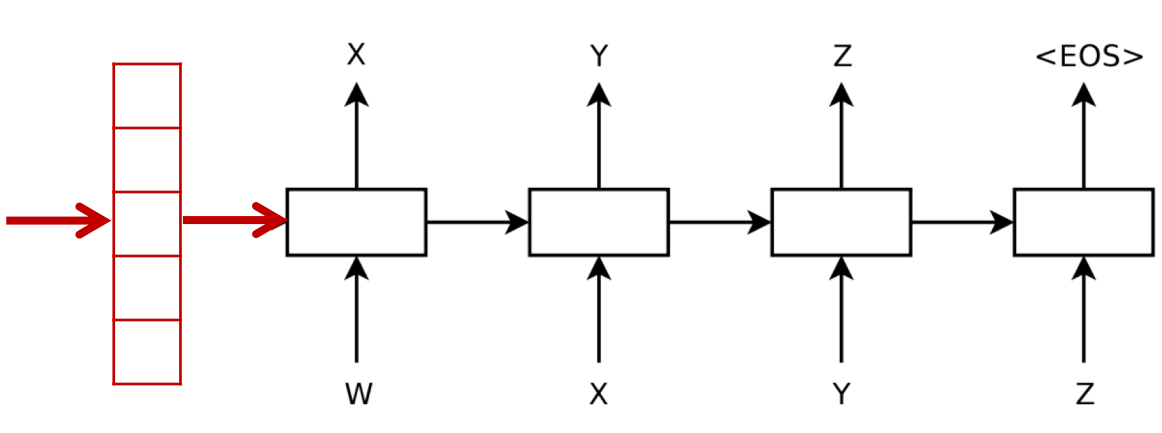
Limitation of Sequence to Sequence Model

Input

The Sequence to Sequence which is

…

…



model is good for Machine Translation

모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

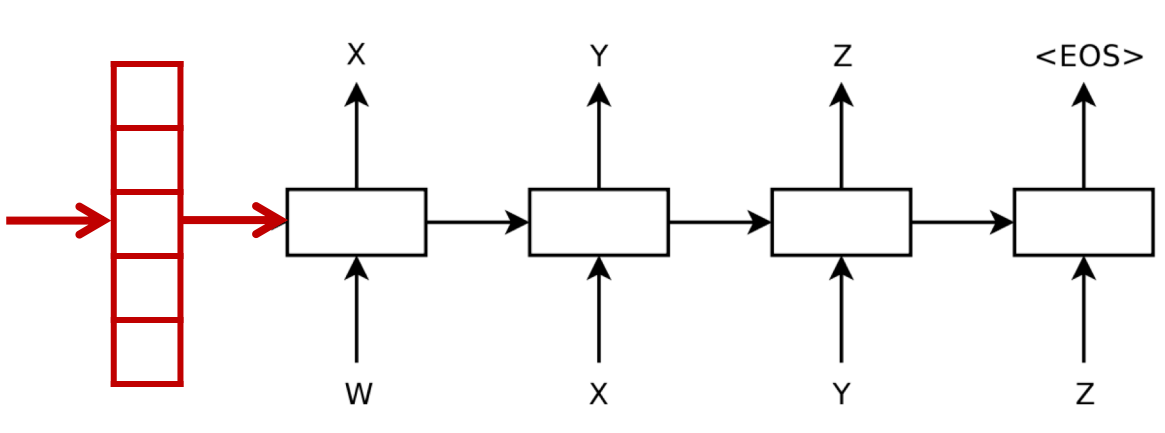
Limitation of Sequence to Sequence Model

Input

The Sequence to Sequence which is

…

…

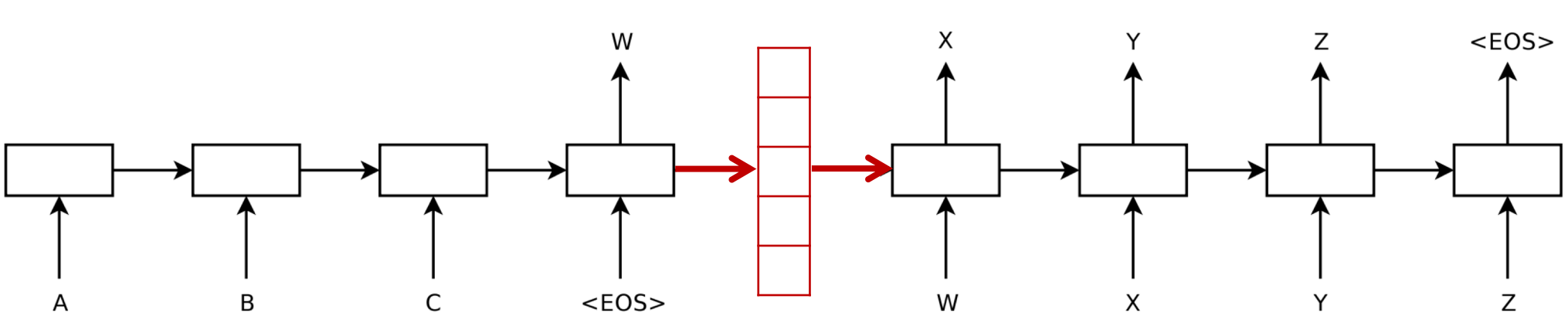


model is good for Machine Translation

모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

Limitation of Sequence to Sequence Model

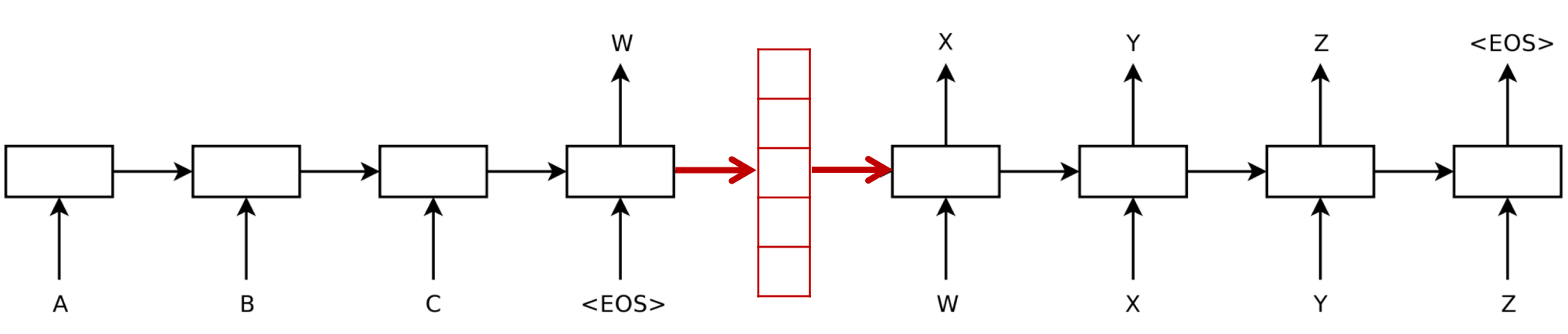


모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

문장이 길어질수록 전체 정보를 하나의 벡터로 표현하기 어려워짐

Why do we need Attention

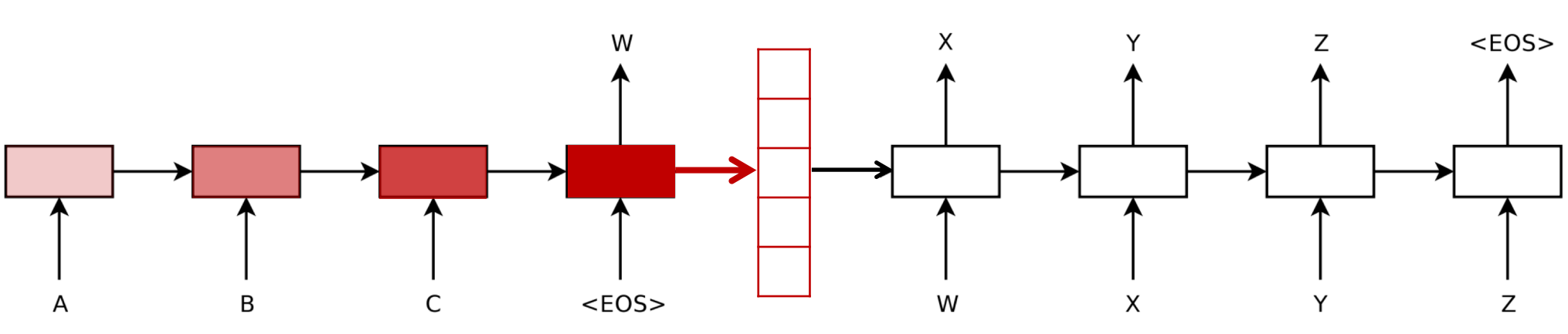
Limitation of Sequence to Sequence Model



모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

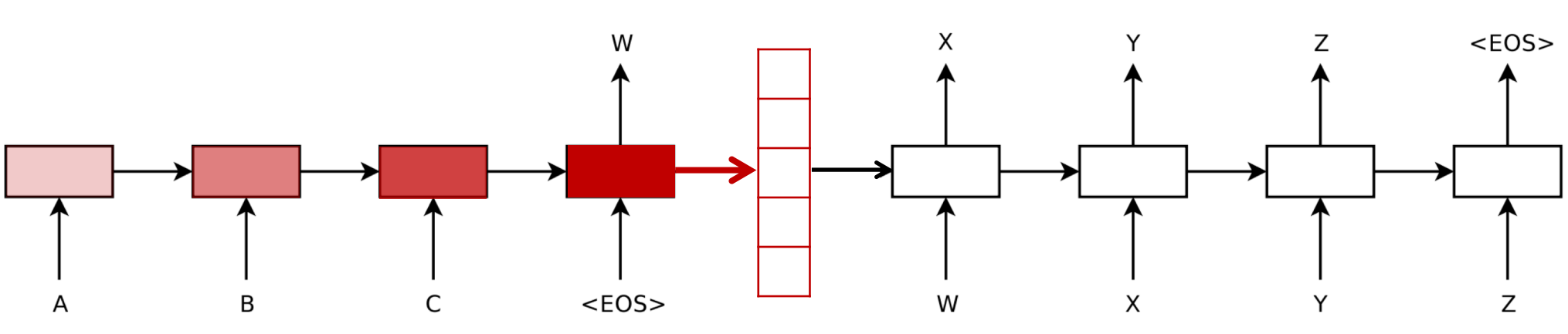
Limitation of Sequence to Sequence Model



모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

Why do we need Attention

Limitation of Sequence to Sequence Model



모든 Input Sequence의 정보를 하나의 fixed-size vector에 압축해서 담고있음

문장 초기 Step의 정보는 마지막 Step에 비해 상대적으로 덜 사용됨

Why do we need Attention

Limitation of Sequence to Sequence Model

- 문장이 길어질수록 전체 정보를 하나의 벡터로 표현하기 어려워짐
- 문장 초기 Step의 정보는 마지막 Step에 비해 상대적으로 덜 사용됨

Why do we need Attention

Limitation of Sequence to Sequence Model

- 문장이 길어질수록 전체 정보를 하나의 벡터로 표현하기 어려워짐
- 문장 초기 Step의 정보는 마지막 Step에 비해 상대적으로 덜 사용됨

⇨ **Neural Machine Translation by Jointly Learning to Align And Translate**

Why do we need Attention

Limitation of Sequence to Sequence Model

- 문장이 길어질수록 전체 정보를 하나의 벡터로 표현하기 어려워짐
- 문장 초기 Step의 정보는 마지막 Step에 비해 상대적으로 덜 사용됨

⇨ **a.k.a Sequence to Sequence with Attention**

Neural Machine Translation By Jointly Learning To Align And Translate

- Bahdanau et al. (2016)
- Attention Mechanism을 처음 도입
- Bidirectional RNN을 사용함

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.

In order to address this issue, we introduce an extension to the encoder–decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

*CIFAR Senior Fellow

Neural Machine Translation By Jointly Learning To Align And Translate

- 문장이 길어질수록 전체 정보를 하나의 벡터로 표현하기 어려워짐
- 문장 초기 Step의 정보는 마지막 Step에 비해 상대적으로 덜 사용됨

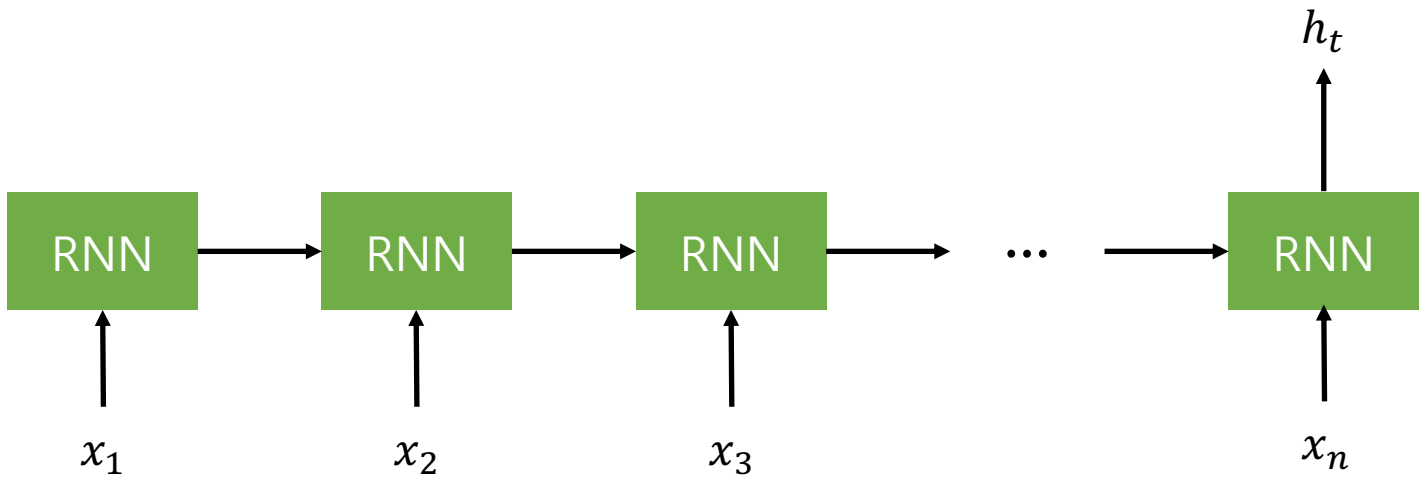
Neural Machine Translation By Jointly Learning To Align And Translate

- 문장이 길어질수록 전체 정보를 하나의 벡터로 표현하기 어려워짐
- 문장 초기 Step의 정보는 마지막 Step에 비해 상대적으로 덜 사용됨

- **Attention Mechanism (alignment model)**
- **Bidirectional RNN**

Neural Machine Translation By Jointly Learning To Align And Translate

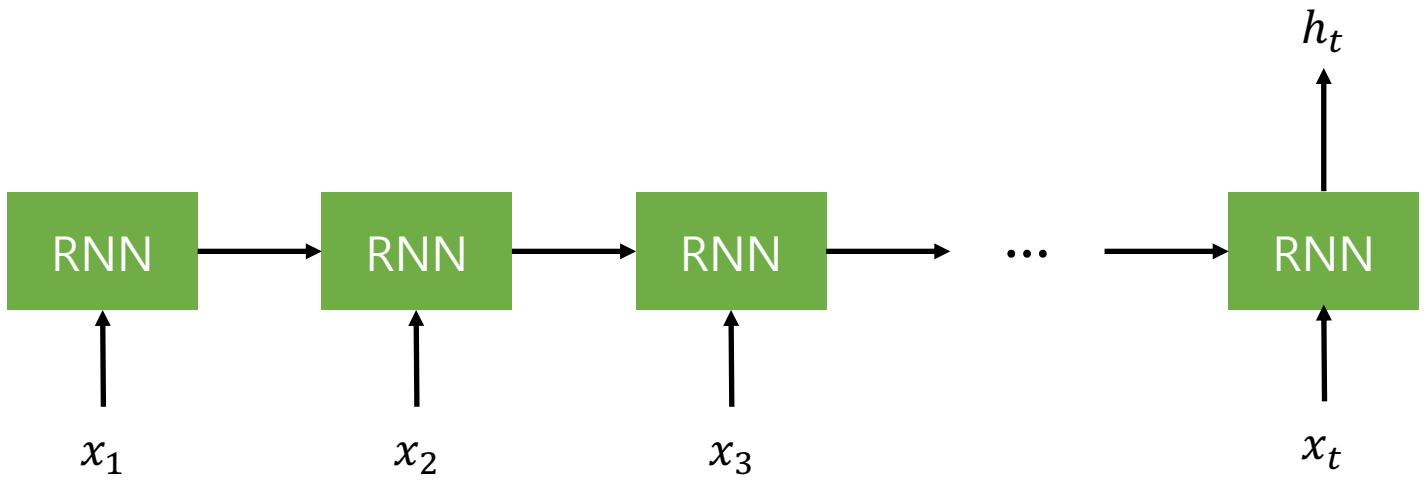
Bidirectional RNN



기존 Unidirectional RNN

Neural Machine Translation By Jointly Learning To Align And Translate

Bidirectional RNN



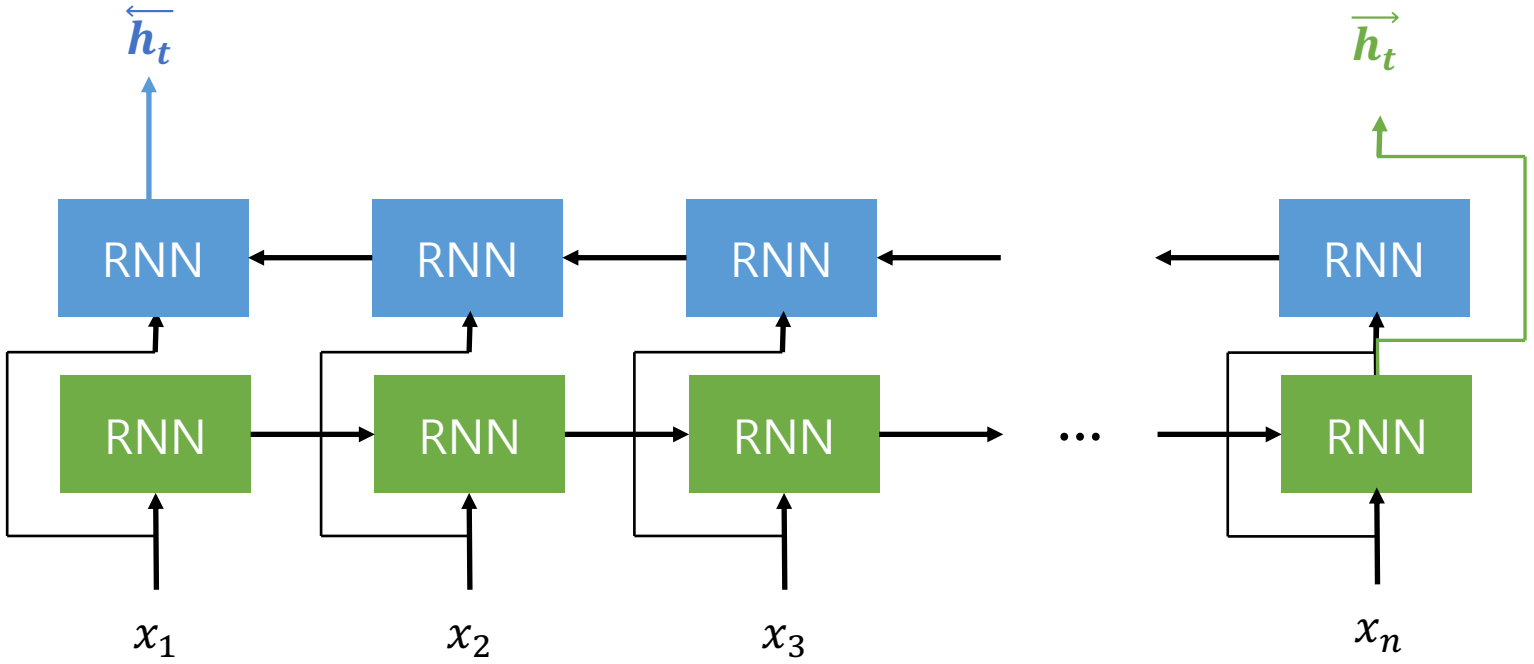
$h_t = f(x_1, x_2, \dots, x_t)$

- f: RNN
- t: Max Length

기존 Unidirectional RNN

Neural Machine Translation By Jointly Learning To Align And Translate

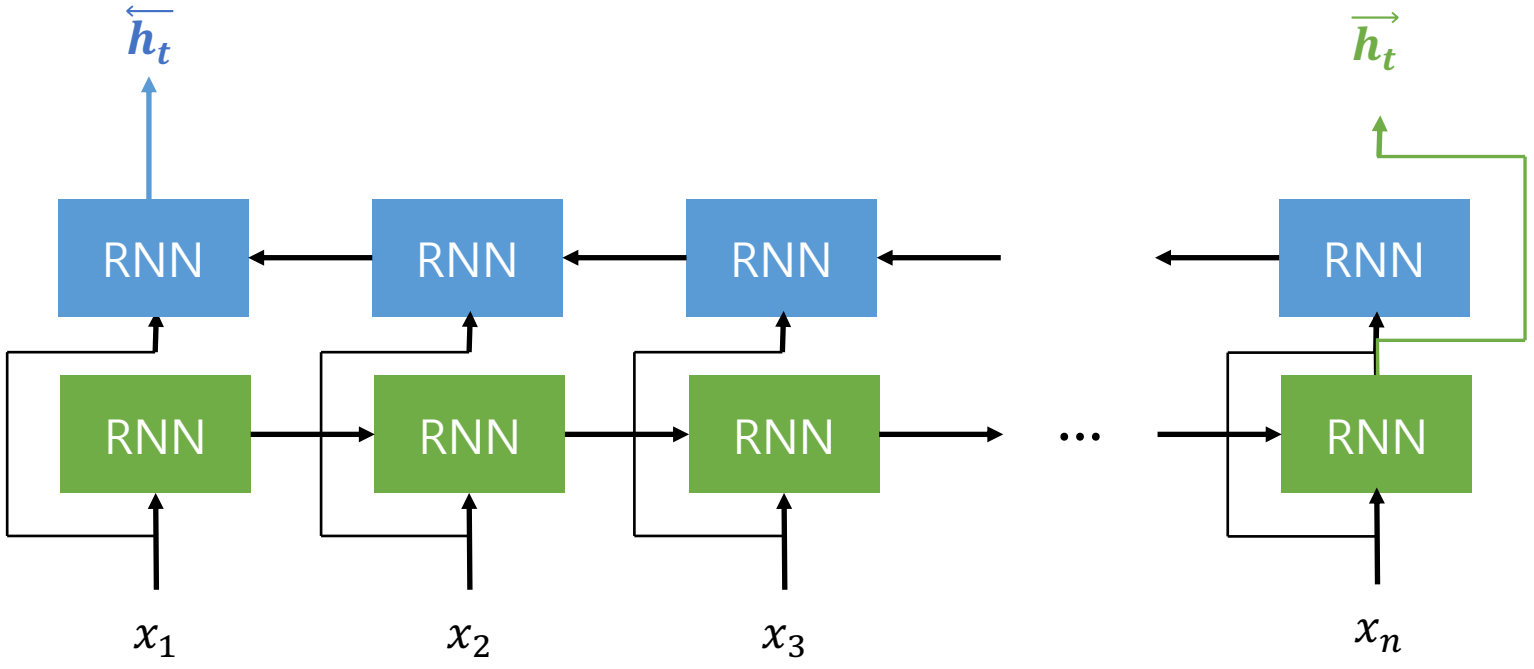
Bidirectional RNN



Bidirectional RNN

Neural Machine Translation By Jointly Learning To Align And Translate

Bidirectional RNN



$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

$$\overrightarrow{h_t} = \vec{f}(x_1, x_2, \dots, x_t)$$

$$\overleftarrow{h_t} = \overleftarrow{f}(x_1, x_2, \dots, x_t)$$

- $\vec{f}$: $Forward\ RNN$
- $\overleftarrow{f}$: $Backward\ RNN$
- t: Max Length

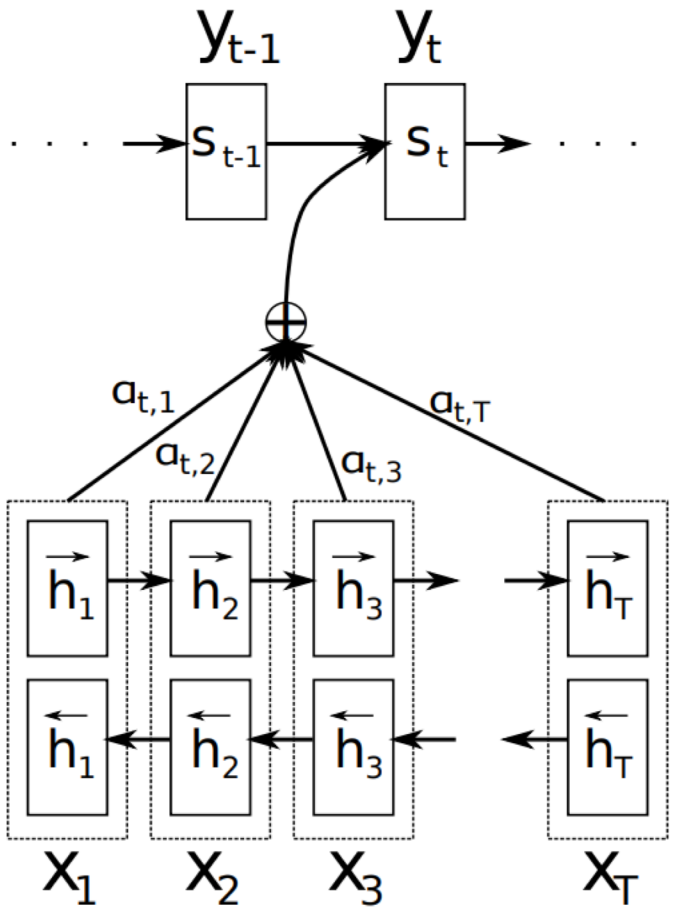
Bidirectional RNN

Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

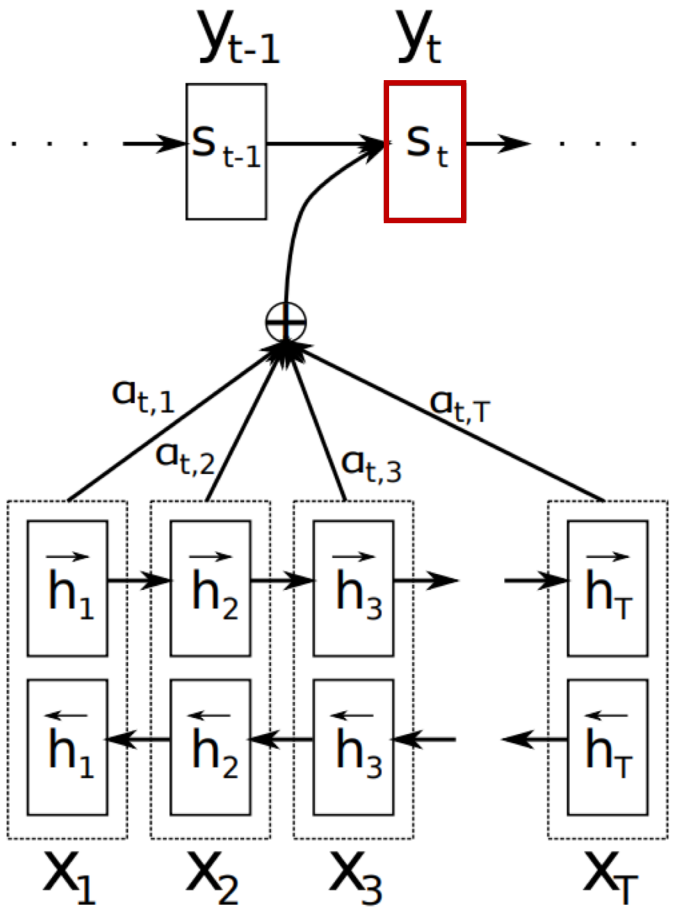
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

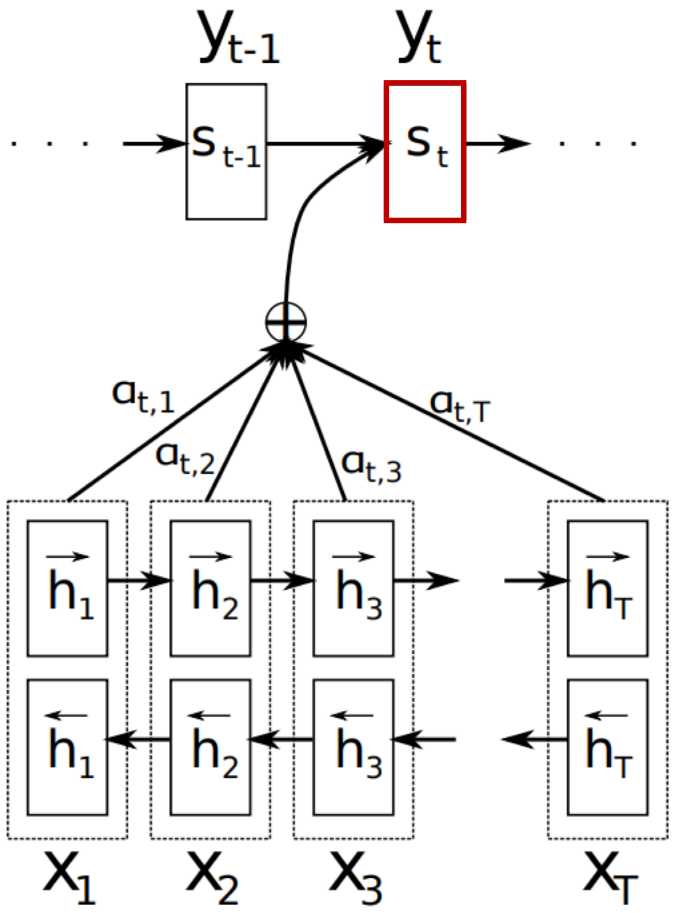


Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$s_t = f(s_{i-1}, y_{i-1}, c_t)$

- s_i: *decoder hidden state*
- y_i: *decoder input*
- c_i: *context vector*

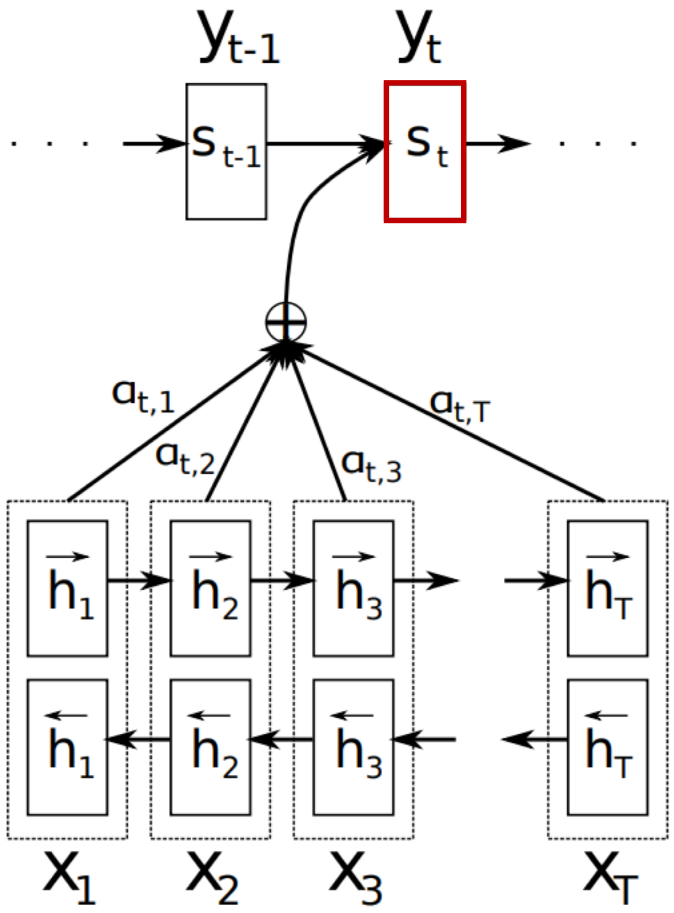


Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$s_t = f(s_{i-1}, y_{i-1}, c_t)$

- s_i: *decoder hidden state*
- y_i: *decoder input*
- c_i: *context vector*

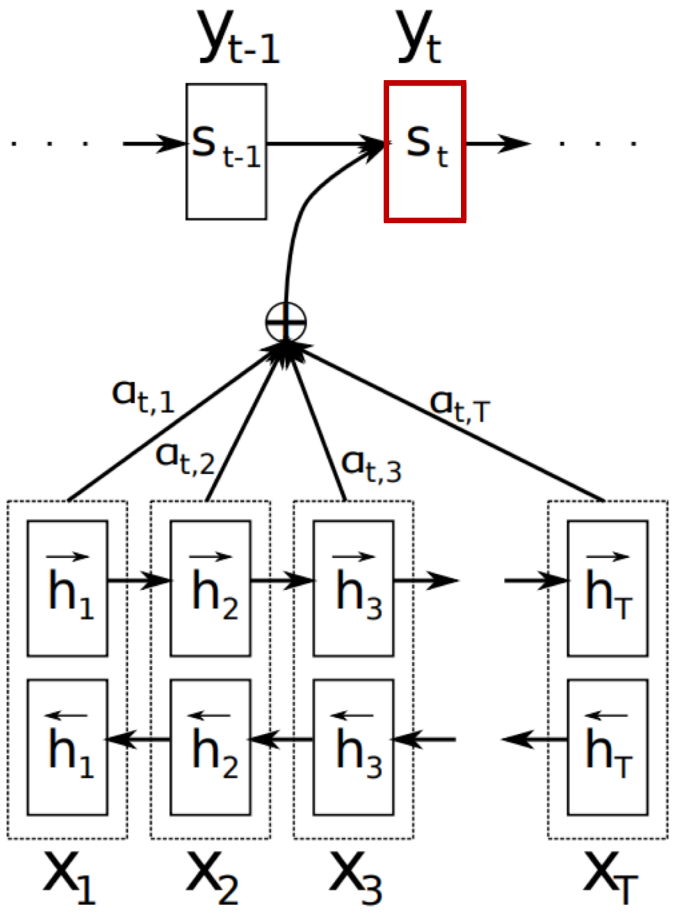


Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

- s_i: *decoder hidden state*
- y_i: *decoder input*
- c_i: *context vector*

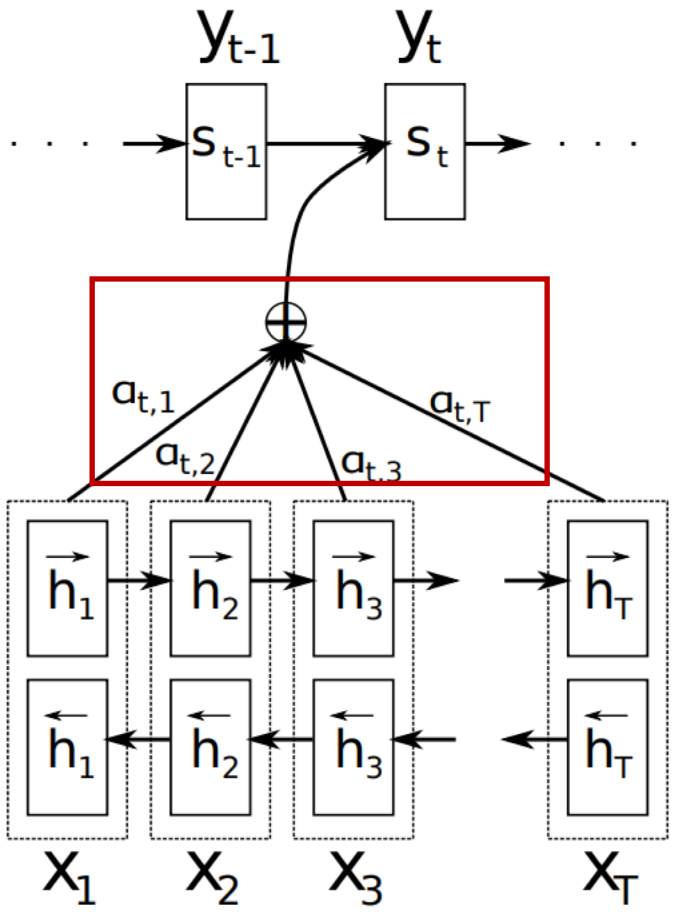


Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

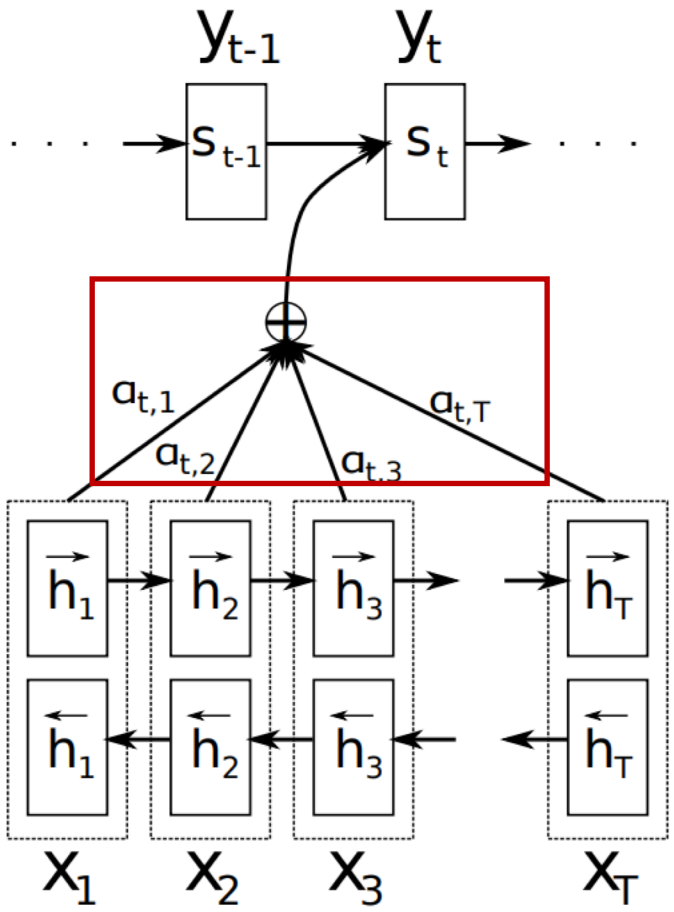


Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$



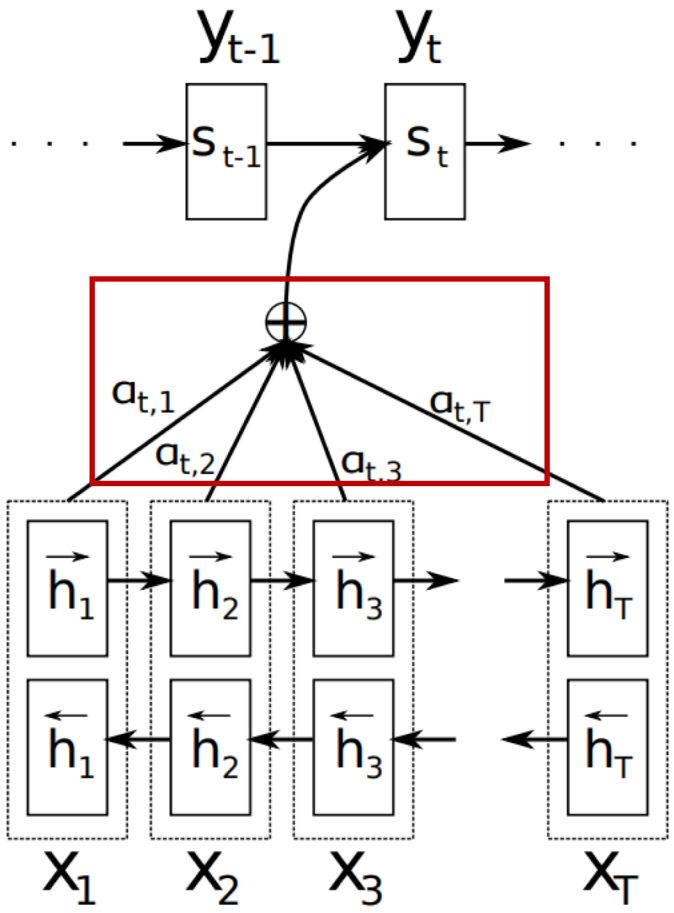
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

attention weight



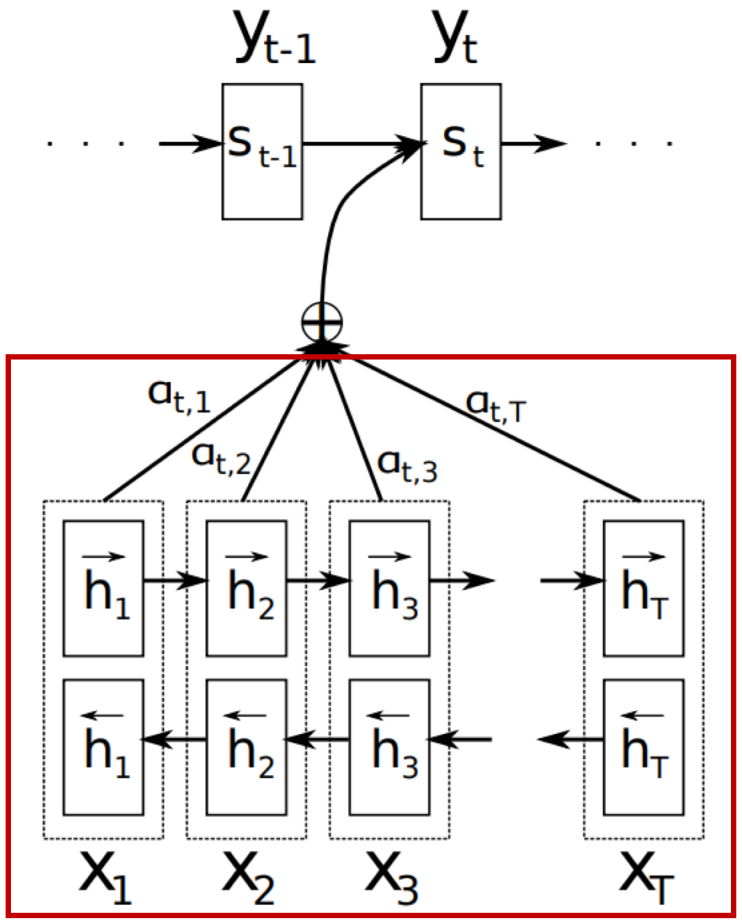
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$



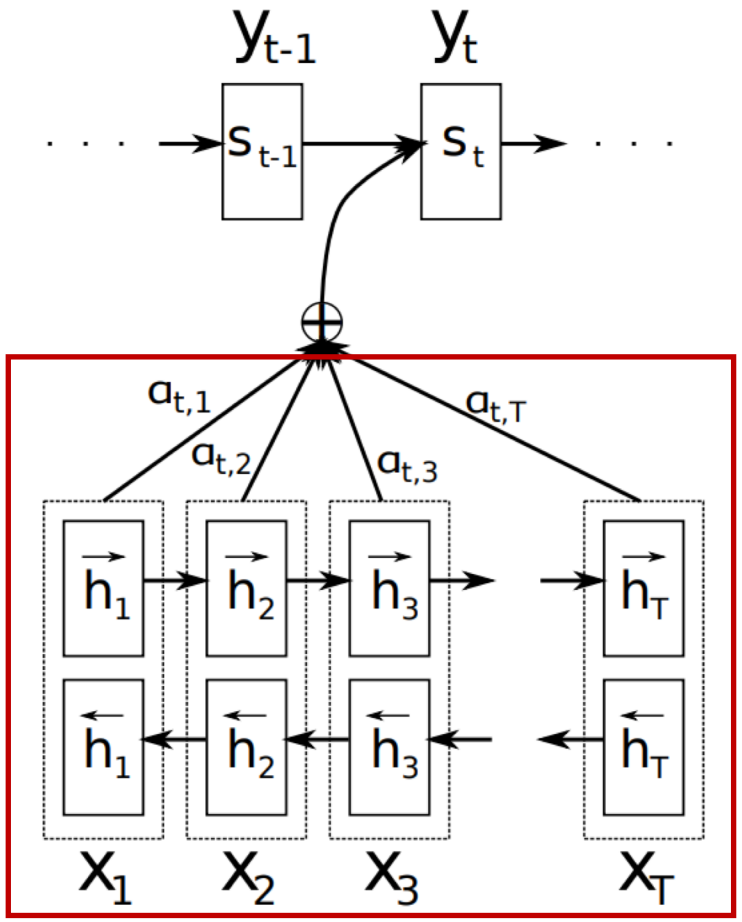
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$



Neural Machine Translation By Jointly Learning To Align And Translate

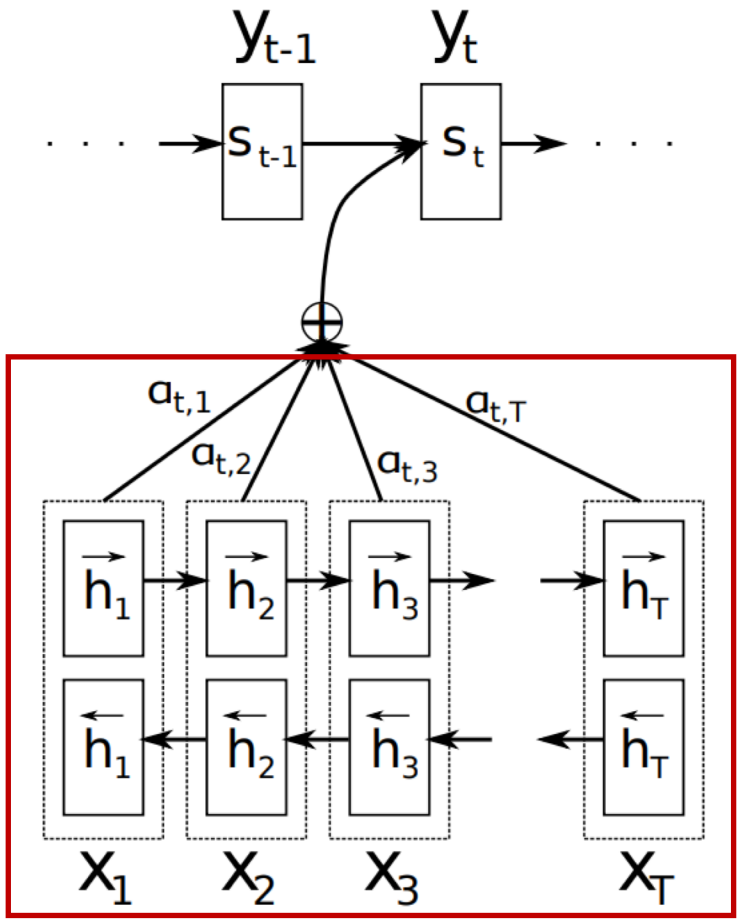
Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

attention score(energy)



Neural Machine Translation By Jointly Learning To Align And Translate

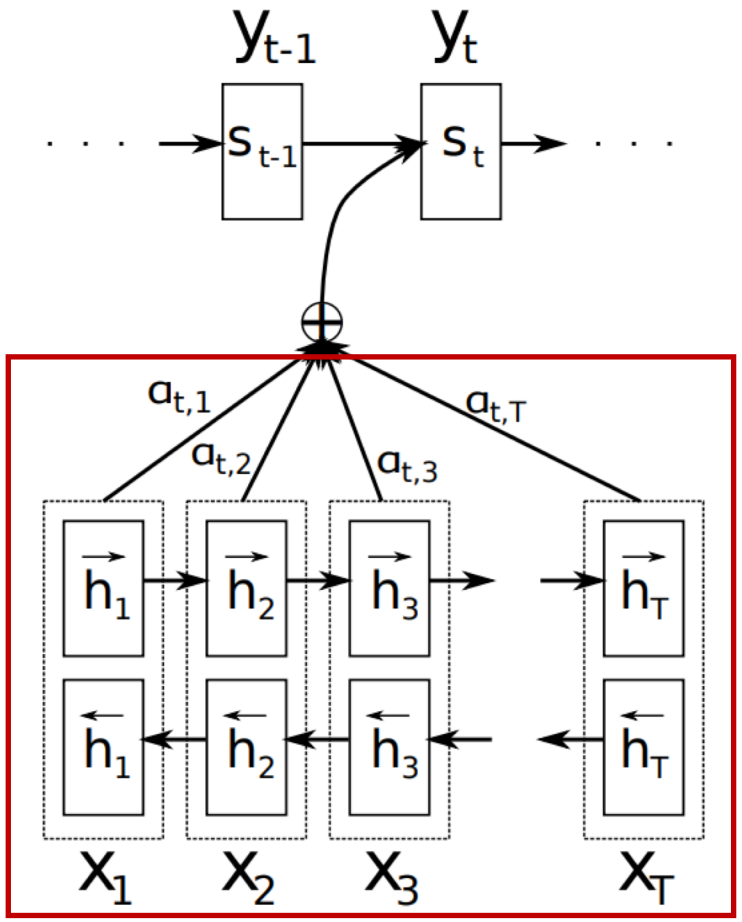
Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$



Neural Machine Translation By Jointly Learning To Align And Translate

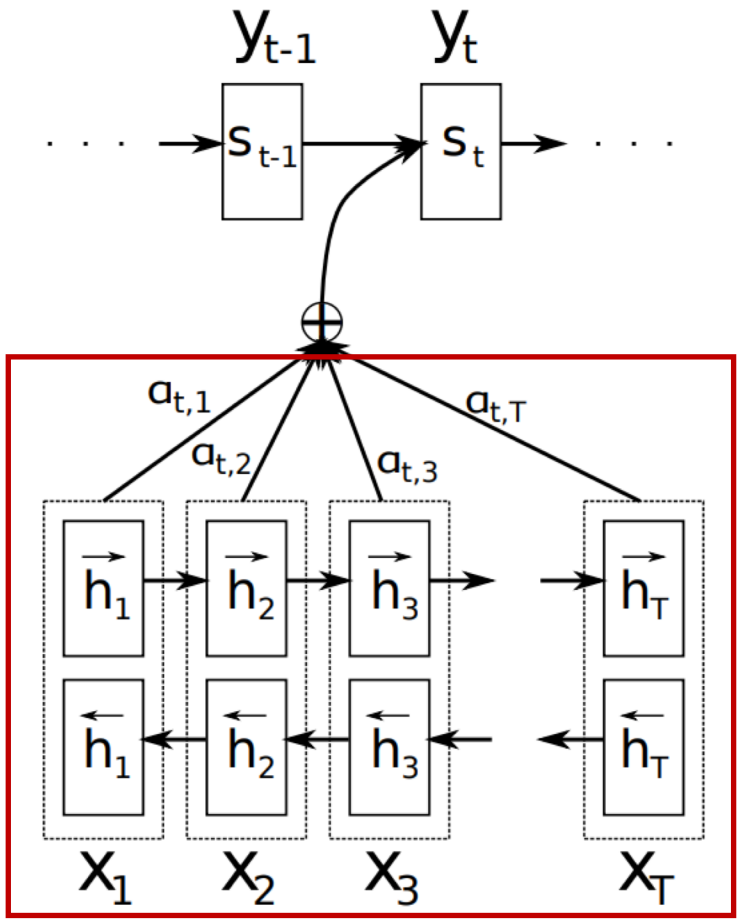
Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

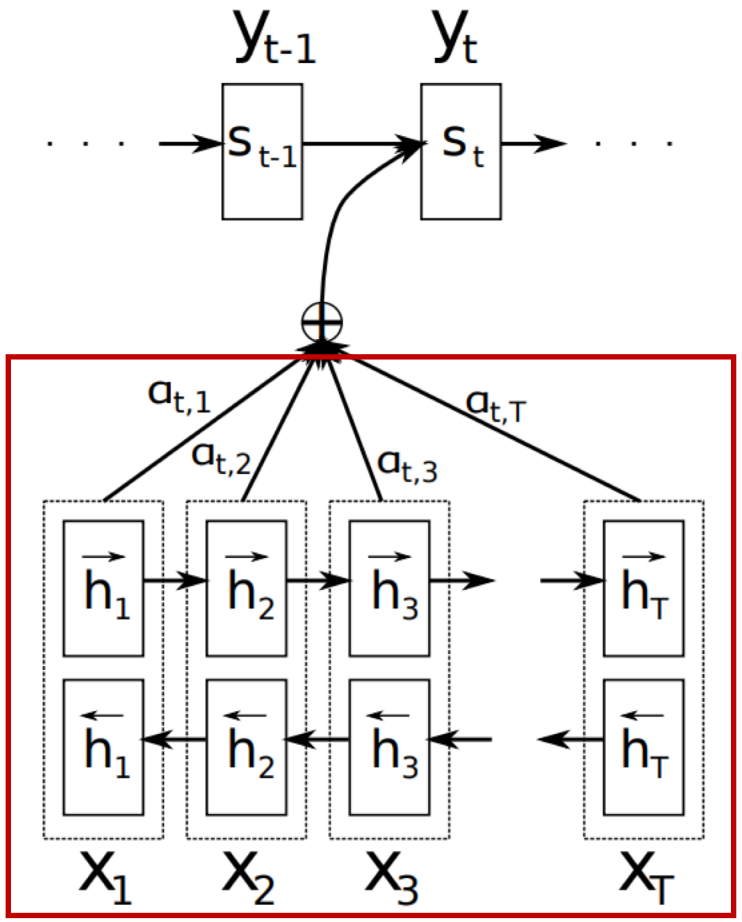
$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

→ j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 중요도



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

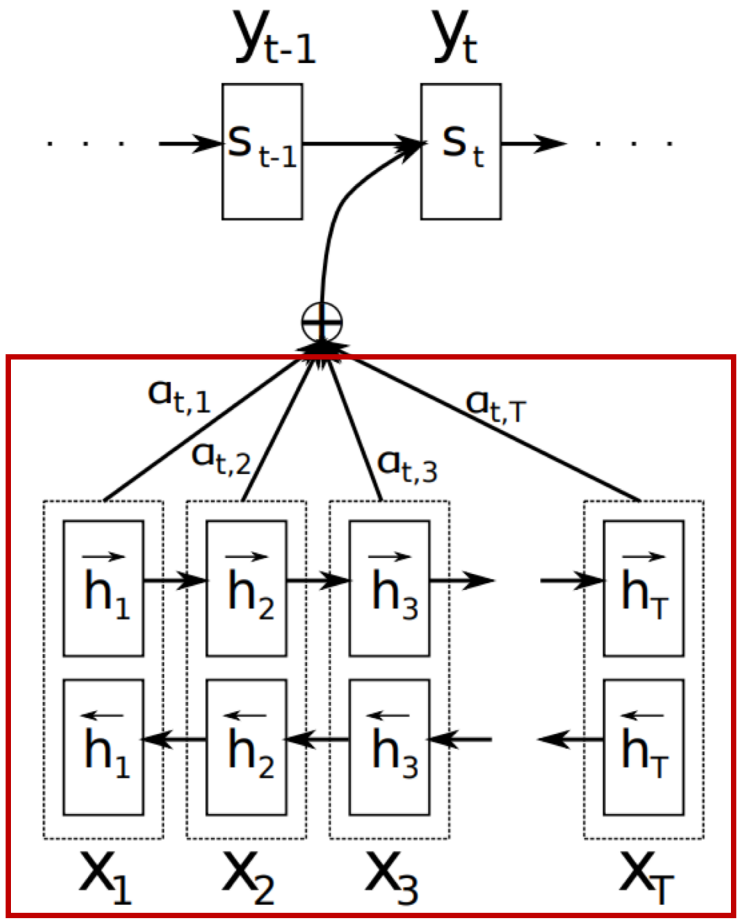
$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 확률

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 중요도



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

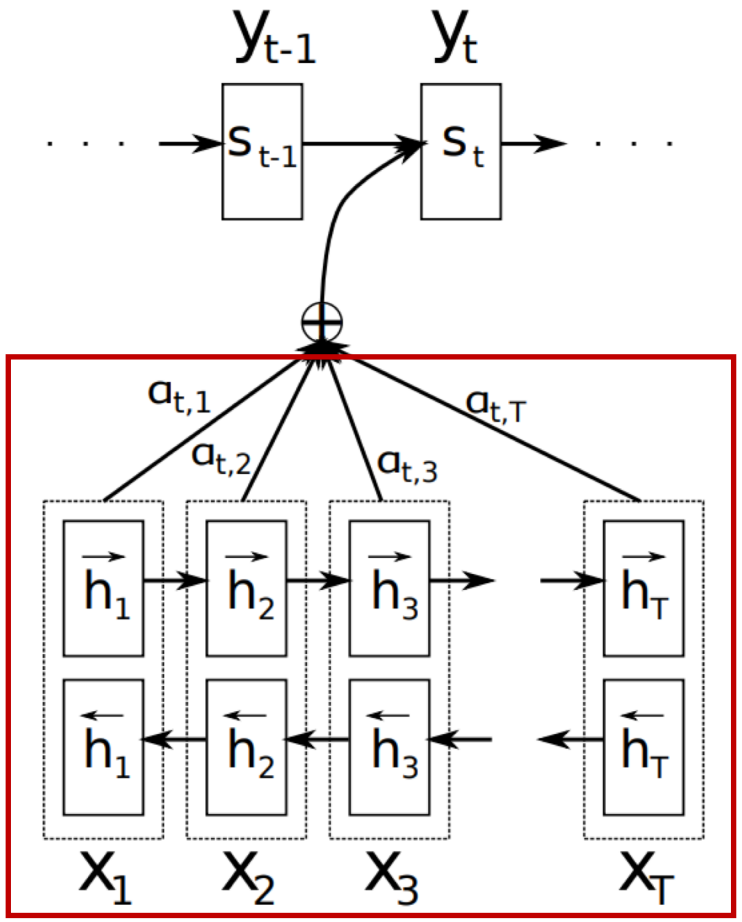
$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 **확률**

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 **중요도**



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

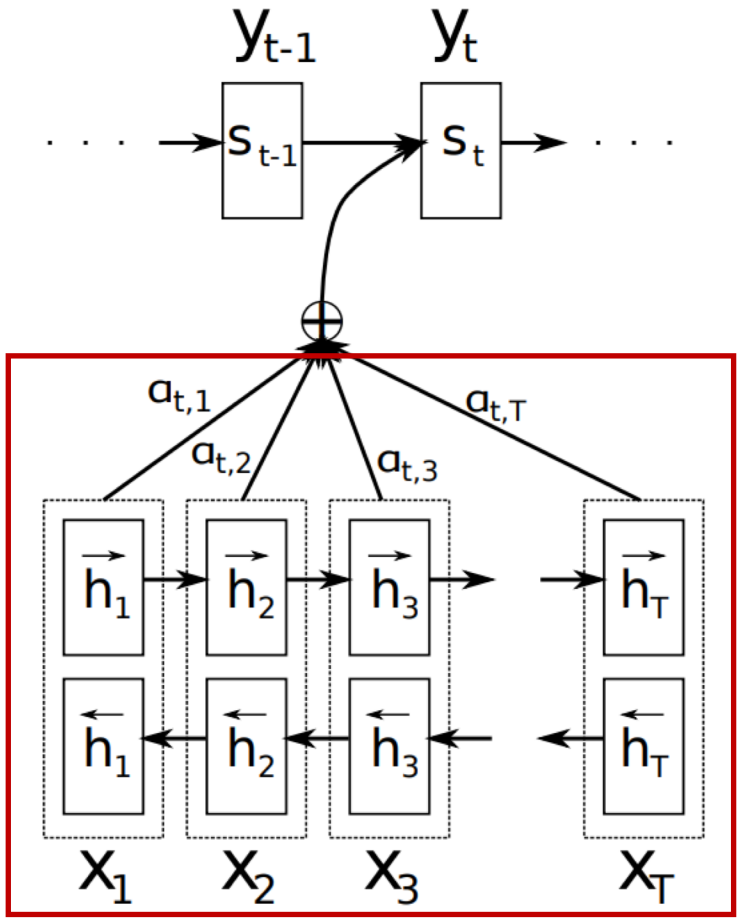
$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 **확률** (0~1 사이값)

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 **중요도**



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

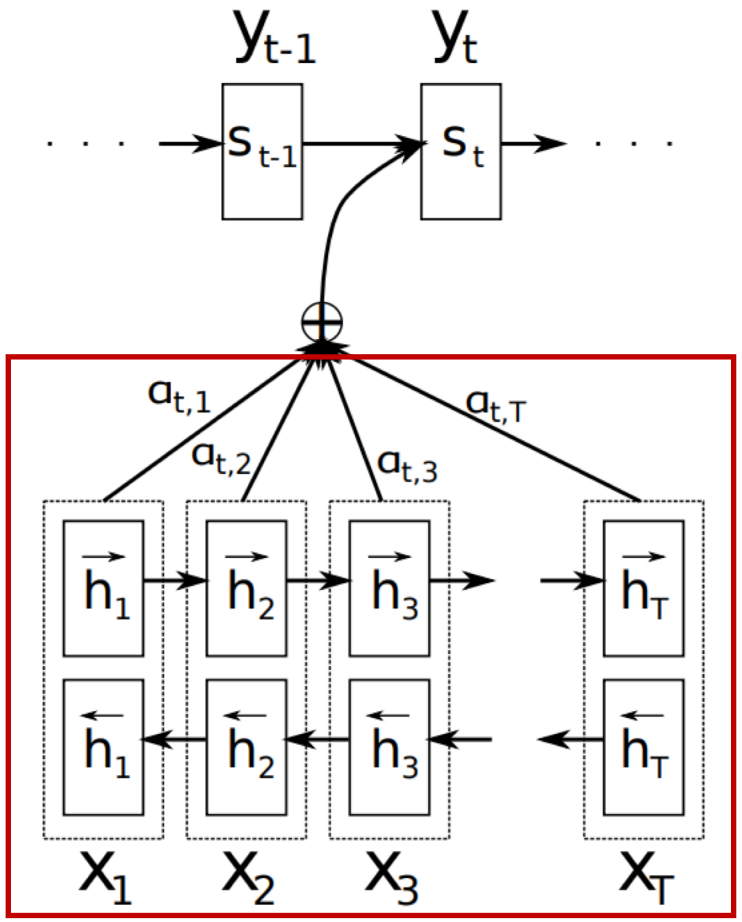
$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 **확률**

j 번째 input을 i 번째 output에 얼마나 반영할지 나타내는 **중요도**



Neural Machine Translation By Jointly Learning To Align And Translate

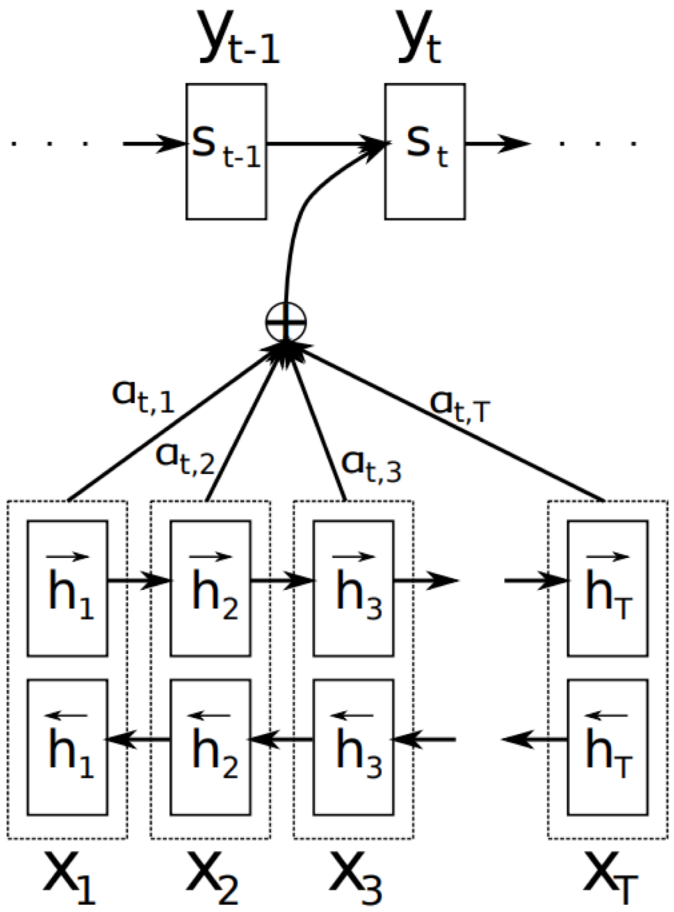
Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

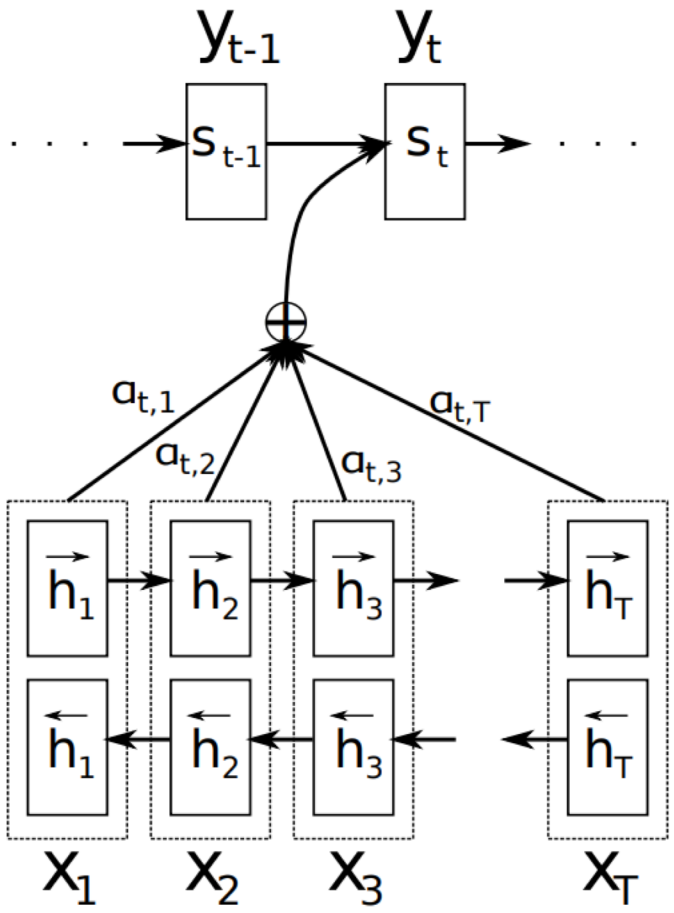
$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

how to calculate attention score?



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

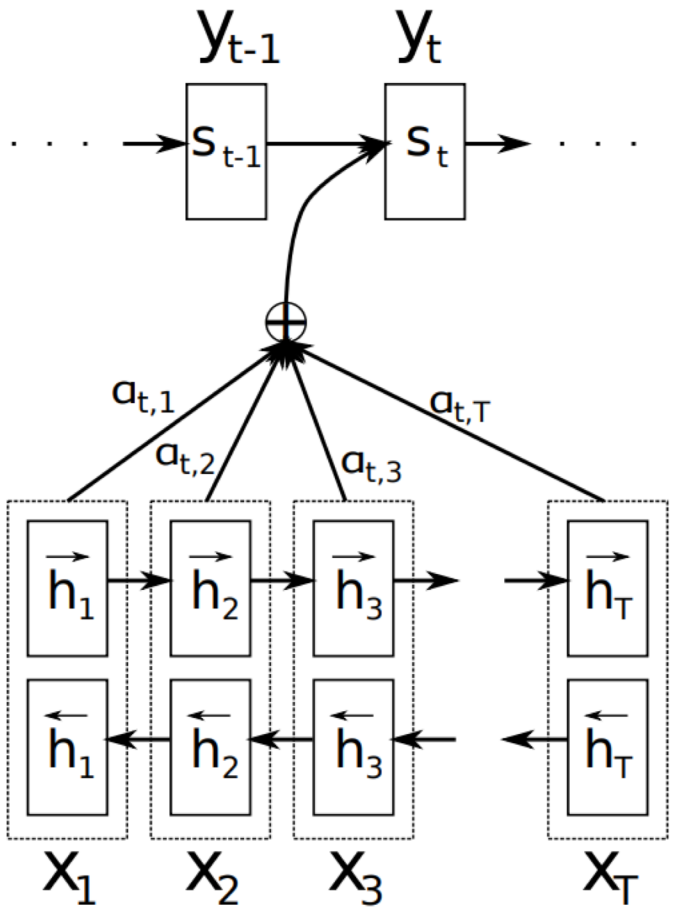
$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

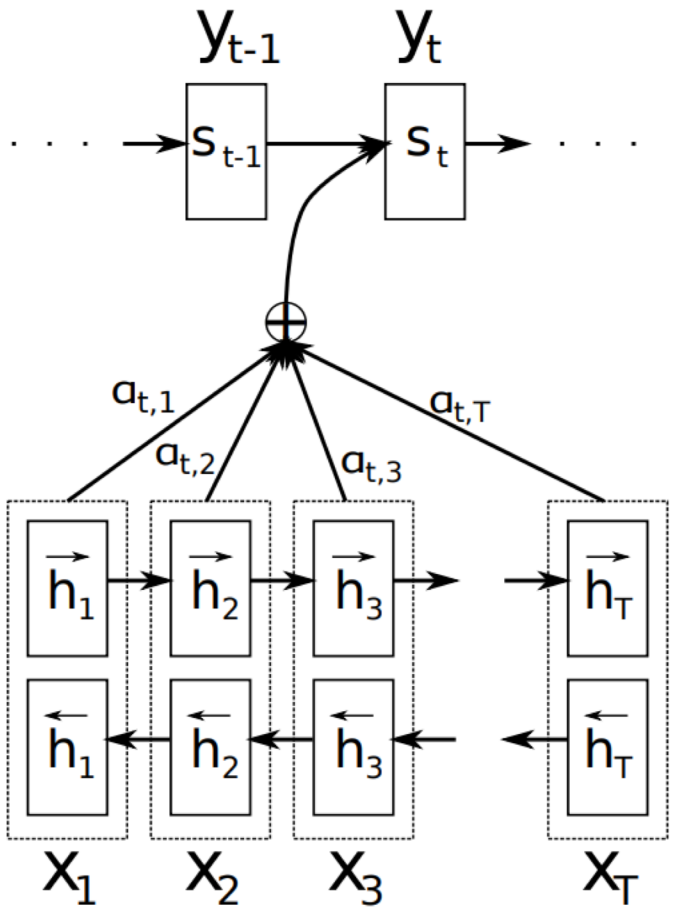
$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Weight matrix

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

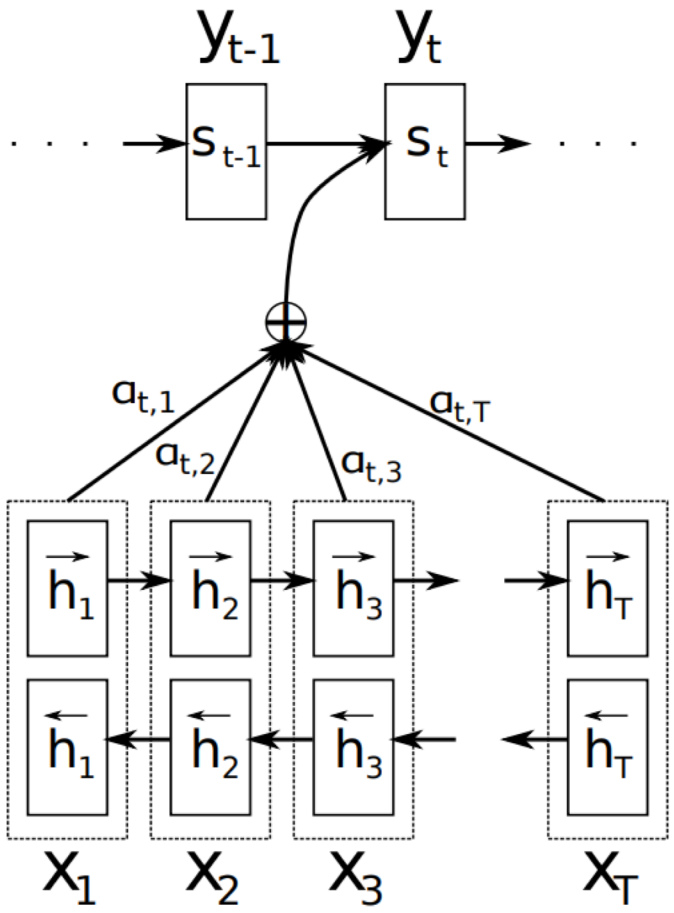
$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

Weight matrix



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

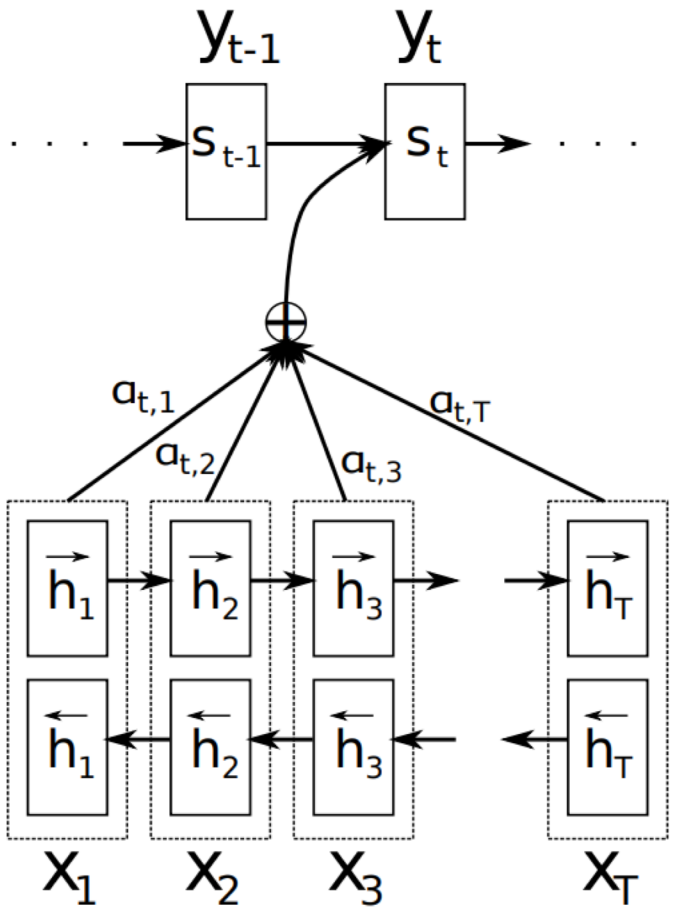
$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

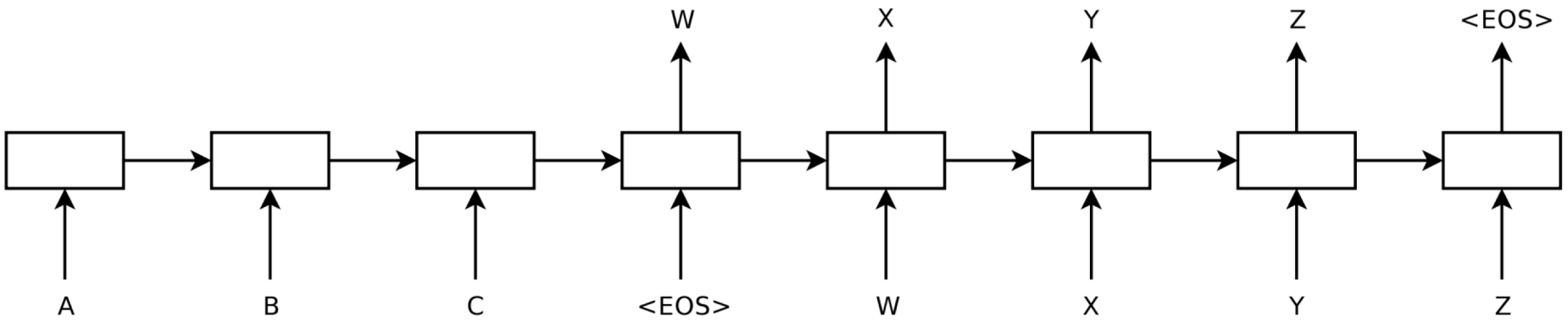
$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

Weight matrix



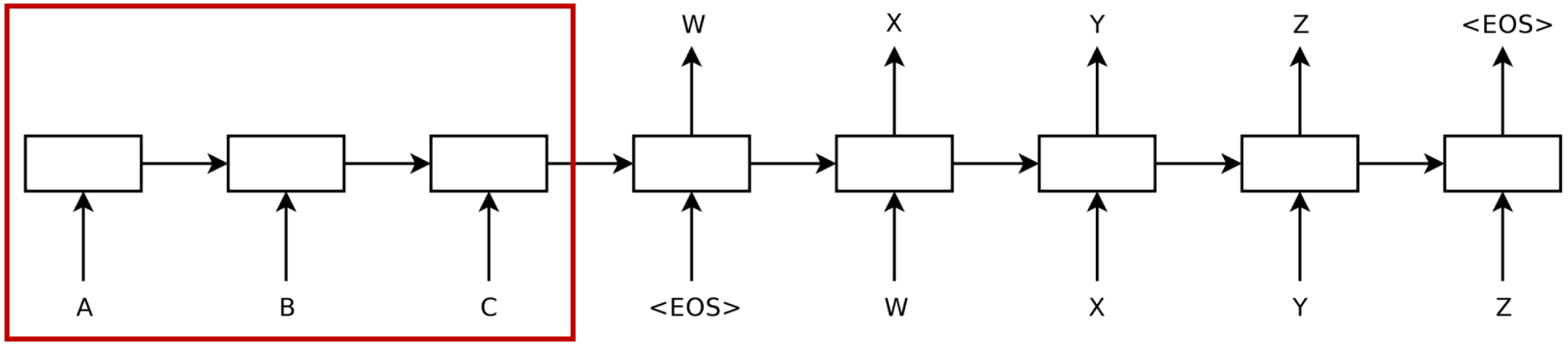
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



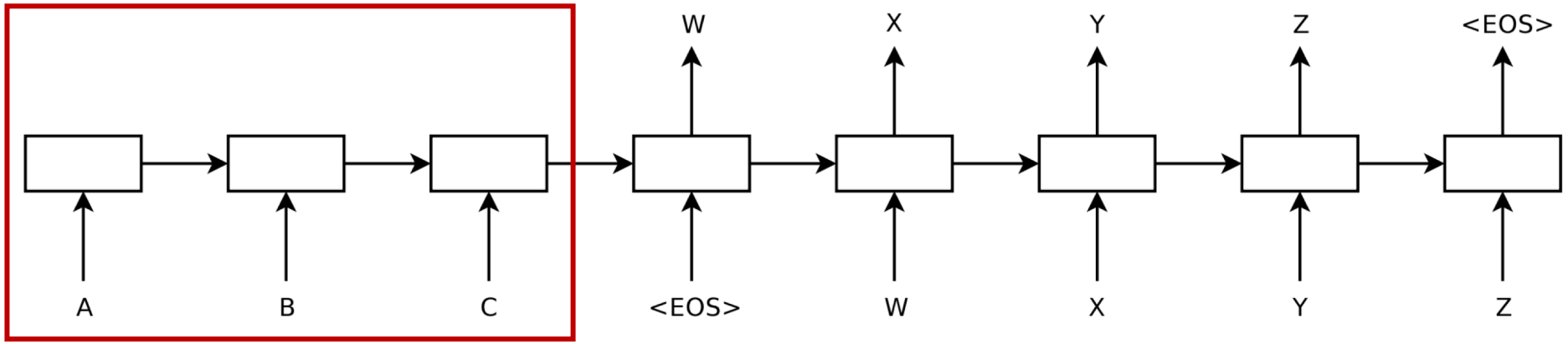
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

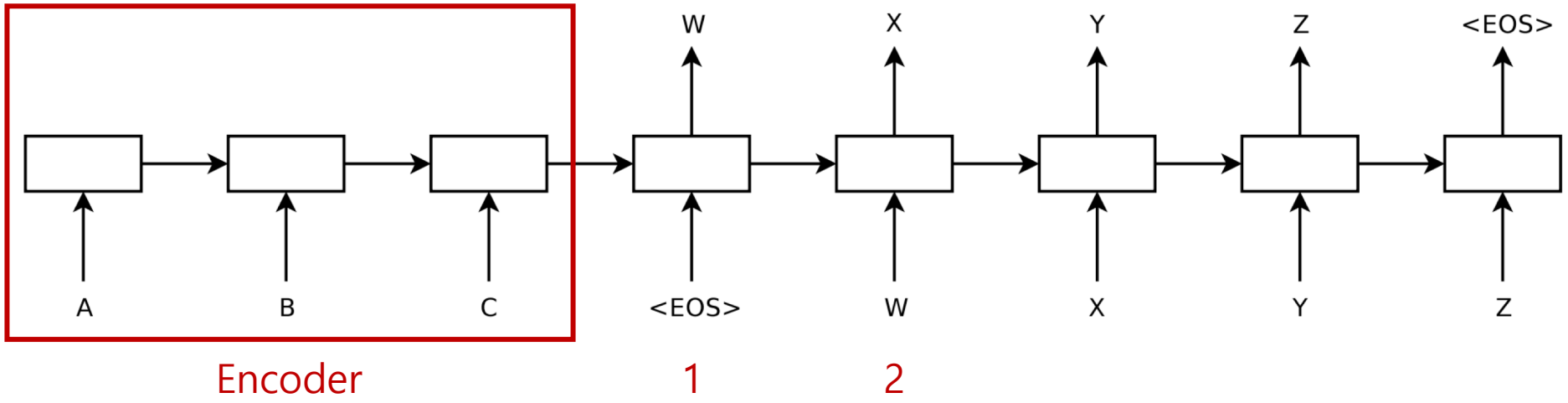


Encoder

1

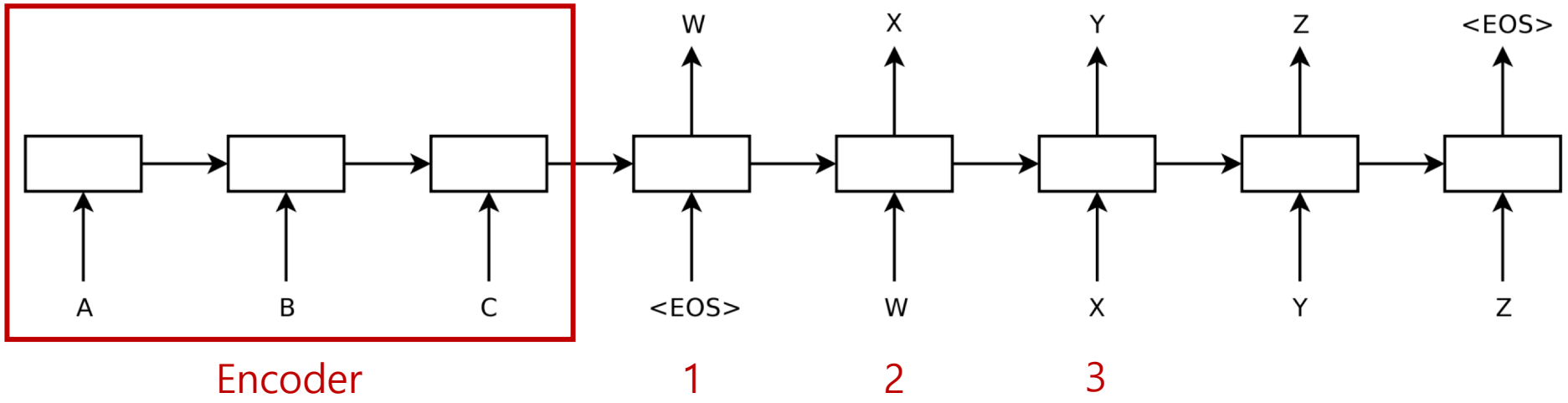
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



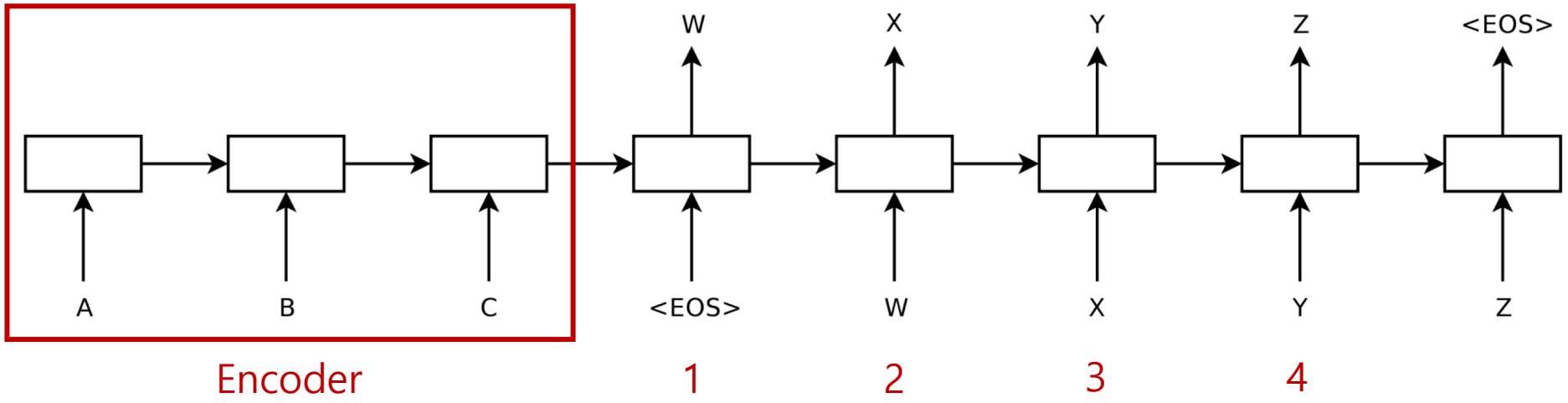
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



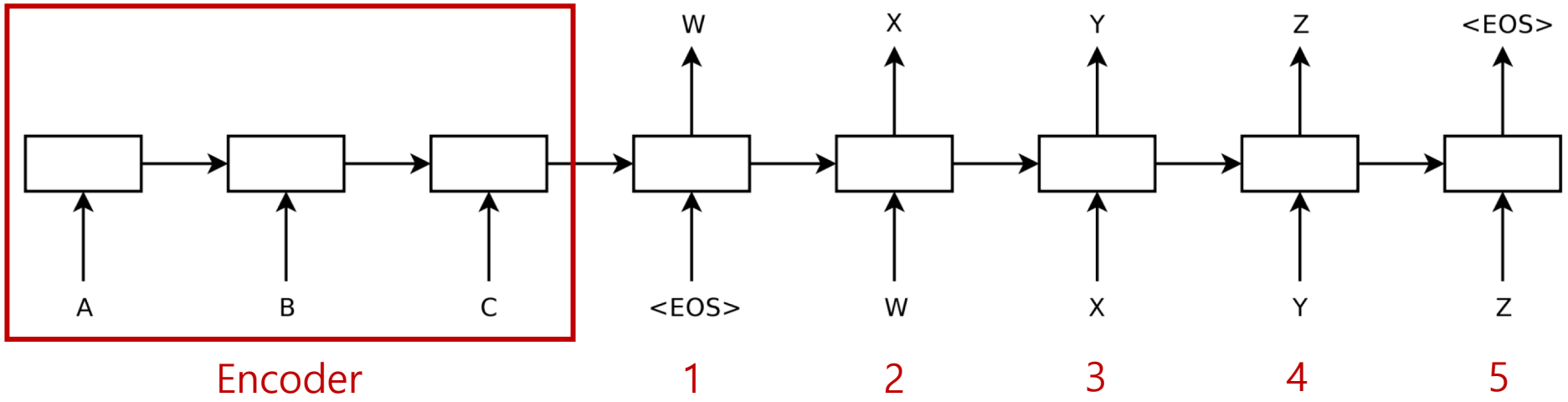
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



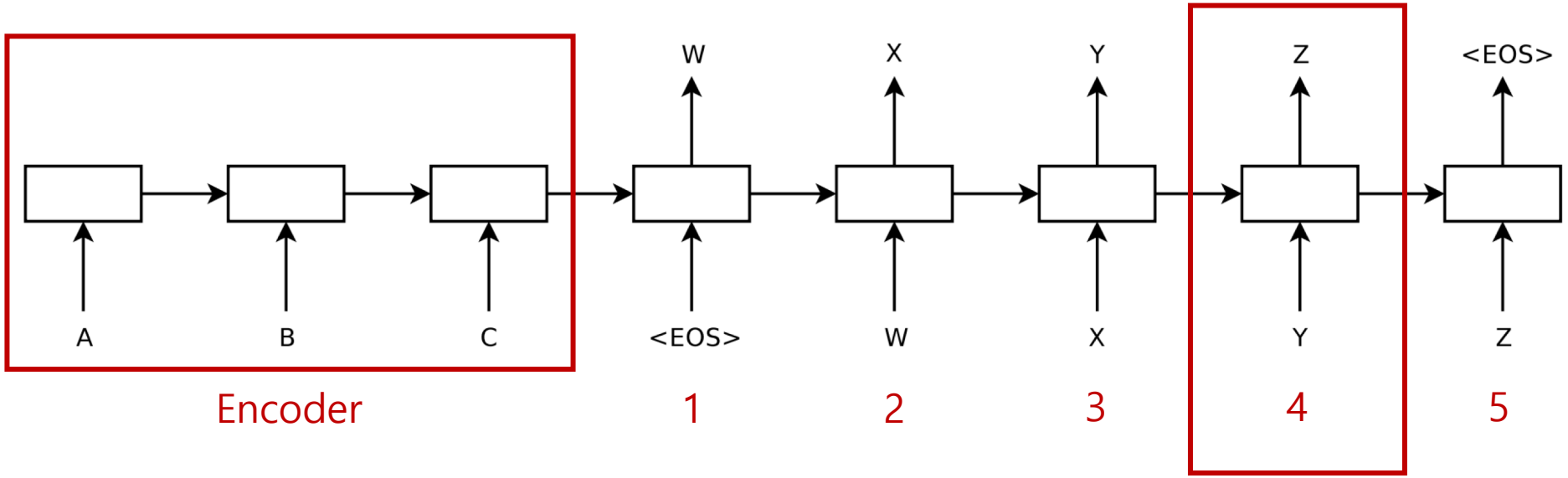
Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)



Neural Machine Translation By Jointly Learning To Align And Translate

Attention Mechanism (alignment model)

Various Attention Mechanism

Various Attention Mechanism

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Additive Attention

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Additive Attention

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

$$= v_a^T \tanh(W_a [s_{i-1}; h_j])$$

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Dot-Product Attention

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Dot-Product Attention

$$a(s_{i-1}, h_j) = s_{i-1} \cdot h_j$$

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Dot-Product Attention

$$a(s_{i-1}, h_j) = s_{i-1} \cdot h_j$$

(s_{i-1} 와 h_j의 dimension이 같아야 함)

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Additive Attention

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$
$$= v_a^T \tanh(W_a [s_{i-1}; h_j])$$

Dot-Product Attention

$$a(s_{i-1}, h_j) = s_{i-1} \cdot h_j$$

Various Attention Mechanism

$$s_t = f(s_{i-1}, y_{i-1}, c_t)$$

$$c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Content-based Attention

Location-based Attention

General Attention

Various Attention Mechanism

Self Attention?

Various Attention Mechanism

Self Attention?

기존의 Attention은 target에 대해서 Source들과의 중요도를 계산

Various Attention Mechanism

Self Attention?

기존의 Attention은 target에 대해서 Source들과의 중요도를 계산

Self Attention은 Source끼리 중요도를 계산

Structured Self-Attentive Sentence Embedding

A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

Zhouhan Lin[‡◇*], **Minwei Feng**[◇], **Cicero Nogueira dos Santos**[◇], **Mo Yu**[◇],
Bing Xiang[◇], **Bowen Zhou**[◇] **& Yoshua Bengio**[‡†]
[◇]IBM Watson
[‡]Montreal Institute for Learning Algorithms (MILA), Université de Montréal
[†]CIFAR Senior Fellow
lin.zhouhan@gmail.com
{mfeng, cicerons, yum, bingxia, zhou}@us.ibm.com

ABSTRACT

This paper proposes a new model for extracting an interpretable sentence embedding by introducing self-attention. Instead of using a vector, we use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. We also propose a self-attention mechanism and a special regularization term for the model. As a side effect, the embedding comes with an easy way of visualizing what specific parts of the sentence are encoded into the embedding. We evaluate our model on 3 different tasks: author profiling, sentiment classification and textual entailment. Results show that our model yields a significant performance gain compared to other sentence embedding methods in all of the 3 tasks.

1 INTRODUCTION

Much progress has been made in learning semantically meaningful distributed representations of individual words, also known as word embeddings (Bengio et al., 2001; Mikolov et al., 2013). On the other hand, much remains to be done to obtain satisfying representations of phrases and sentences. Those methods generally fall into two categories. The first consists of universal sentence embeddings usually trained by unsupervised learning (Hill et al., 2016). This includes SkipThought vectors (Kiros et al., 2015), ParagraphVector (Le & Mikolov, 2014), recursive auto-encoders (Socher et al., 2011; 2013), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), etc.

The other category consists of models trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. One generally finds that specifically trained sentence embeddings perform better than generic ones, although generic ones can be used in a semi-supervised setting, exploiting large unlabeled corpora. Several models have been proposed along this line, by using recurrent networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), recursive networks (Socher et al., 2013) and convolutional networks (Kalchbrenner et al., 2014; dos Santos & Gatti, 2014; Kim, 2014) as an intermediate step in creating sentence representations to solve a wide variety of tasks including classification and ranking (Yin & Schütze, 2015; Palangi et al., 2016; Tan et al., 2016; Feng et al., 2015). A common approach in previous methods consists in creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. Additional works have also been done in exploiting linguistic structures such as parse and dependence trees to improve sentence representations (Ma et al., 2015; Mou et al., 2015b; Tai et al., 2015).

For some tasks people propose to use attention mechanism on top of the CNN or LSTM model to introduce extra source of information to guide the extraction of sentence embedding (dos Santos et al., 2016). However, for some other tasks like sentiment classification, this is not directly applicable since there is no such extra information: the model is only given one single sentence as input. In those cases, the most common way is to add a max pooling or averaging step across all time steps

[*]This work has been done during the 1st author's internship with IBM Watson.

Structured Self-Attentive Sentence Embedding

- Zhouhan et al. (2016)

A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

Zhouhan Lin[‡◇*], **Minwei Feng**[◇], **Cicero Nogueira dos Santos**[◇], **Mo Yu**[◇],
Bing Xiang[◇], **Bowen Zhou**[◇] **& Yoshua Bengio**[‡†]
[◇]IBM Watson
[‡]Montreal Institute for Learning Algorithms (MILA), Université de Montréal
[†]CIFAR Senior Fellow
lin.zhouhan@gmail.com
{mfeng, cicerons, yum, bingxia, zhou}@us.ibm.com

ABSTRACT

This paper proposes a new model for extracting an interpretable sentence embedding by introducing self-attention. Instead of using a vector, we use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. We also propose a self-attention mechanism and a special regularization term for the model. As a side effect, the embedding comes with an easy way of visualizing what specific parts of the sentence are encoded into the embedding. We evaluate our model on 3 different tasks: author profiling, sentiment classification and textual entailment. Results show that our model yields a significant performance gain compared to other sentence embedding methods in all of the 3 tasks.

1 INTRODUCTION

Much progress has been made in learning semantically meaningful distributed representations of individual words, also known as word embeddings (Bengio et al., 2001; Mikolov et al., 2013). On the other hand, much remains to be done to obtain satisfying representations of phrases and sentences. Those methods generally fall into two categories. The first consists of universal sentence embeddings usually trained by unsupervised learning (Hill et al., 2016). This includes SkipThought vectors (Kiros et al., 2015), ParagraphVector (Le & Mikolov, 2014), recursive auto-encoders (Socher et al., 2011; 2013), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), etc.

The other category consists of models trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. One generally finds that specifically trained sentence embeddings perform better than generic ones, although generic ones can be used in a semi-supervised setting, exploiting large unlabeled corpora. Several models have been proposed along this line, by using recurrent networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), recursive networks (Socher et al., 2013) and convolutional networks (Kalchbrenner et al., 2014; dos Santos & Gatti, 2014; Kim, 2014) as an intermediate step in creating sentence representations to solve a wide variety of tasks including classification and ranking (Yin & Schütze, 2015; Palangi et al., 2016; Tan et al., 2016; Feng et al., 2015). A common approach in previous methods consists in creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. Additional works have also been done in exploiting linguistic structures such as parse and dependence trees to improve sentence representations (Ma et al., 2015; Mou et al., 2015b; Tai et al., 2015).

For some tasks people propose to use attention mechanism on top of the CNN or LSTM model to introduce extra source of information to guide the extraction of sentence embedding (dos Santos et al., 2016). However, for some other tasks like sentiment classification, this is not directly applicable since there is no such extra information: the model is only given one single sentence as input. In those cases, the most common way is to add a max pooling or averaging step across all time steps

[*]This work has been done during the 1st author's internship with IBM Watson.

Structured Self-Attentive Sentence Embedding

- Zhouhan et al. (2016)
- Sentence 단위를 Embedding하기 위한 기법

A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

Zhouhan Lin[‡◇*], **Minwei Feng**[◇], **Cicero Nogueira dos Santos**[◇], **Mo Yu**[◇],
Bing Xiang[◇], **Bowen Zhou**[◇] **& Yoshua Bengio**[‡†]
[◇]IBM Watson
[‡]Montreal Institute for Learning Algorithms (MILA), Université de Montréal
[†]CIFAR Senior Fellow
lin.zhouhan@gmail.com
{mfeng, cicerons, yum, bingxia, zhou}@us.ibm.com

ABSTRACT

This paper proposes a new model for extracting an interpretable sentence embedding by introducing self-attention. Instead of using a vector, we use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. We also propose a self-attention mechanism and a special regularization term for the model. As a side effect, the embedding comes with an easy way of visualizing what specific parts of the sentence are encoded into the embedding. We evaluate our model on 3 different tasks: author profiling, sentiment classification and textual entailment. Results show that our model yields a significant performance gain compared to other sentence embedding methods in all of the 3 tasks.

1 INTRODUCTION

Much progress has been made in learning semantically meaningful distributed representations of individual words, also known as word embeddings (Bengio et al., 2001; Mikolov et al., 2013). On the other hand, much remains to be done to obtain satisfying representations of phrases and sentences. Those methods generally fall into two categories. The first consists of universal sentence embeddings usually trained by unsupervised learning (Hill et al., 2016). This includes SkipThought vectors (Kiros et al., 2015), ParagraphVector (Le & Mikolov, 2014), recursive auto-encoders (Socher et al., 2011; 2013), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), etc.

The other category consists of models trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. One generally finds that specifically trained sentence embeddings perform better than generic ones, although generic ones can be used in a semi-supervised setting, exploiting large unlabeled corpora. Several models have been proposed along this line, by using recurrent networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), recursive networks (Socher et al., 2013) and convolutional networks (Kalchbrenner et al., 2014; dos Santos & Gatti, 2014; Kim, 2014) as an intermediate step in creating sentence representations to solve a wide variety of tasks including classification and ranking (Yin & Schütze, 2015; Palangi et al., 2016; Tan et al., 2016; Feng et al., 2015). A common approach in previous methods consists in creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. Additional works have also been done in exploiting linguistic structures such as parse and dependence trees to improve sentence representations (Ma et al., 2015; Mou et al., 2015b; Tai et al., 2015).

For some tasks people propose to use attention mechanism on top of the CNN or LSTM model to introduce extra source of information to guide the extraction of sentence embedding (dos Santos et al., 2016). However, for some other tasks like sentiment classification, this is not directly applicable since there is no such extra information: the model is only given one single sentence as input. In those cases, the most common way is to add a max pooling or averaging step across all time steps

[*]This work has been done during the 1st author's internship with IBM Watson.

Structured Self-Attentive Sentence Embedding

- Zhouhan et al. (2016)
- Sentence 단위를 Embedding하기 위한 기법
- Sentence 내의 관계를 파악하기 위해 Self Attention 사용

A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

Zhouhan Lin[‡◇*], **Minwei Feng**[◇], **Cicero Nogueira dos Santos**[◇], **Mo Yu**[◇],
Bing Xiang[◇], **Bowen Zhou**[◇] **& Yoshua Bengio**[‡†]
[◇]IBM Watson
[‡]Montreal Institute for Learning Algorithms (MILA), Université de Montréal
[†]CIFAR Senior Fellow
lin.zhouhan@gmail.com
{mfeng, cicerons, yum, bingxia, zhou}@us.ibm.com

ABSTRACT

This paper proposes a new model for extracting an interpretable sentence embedding by introducing self-attention. Instead of using a vector, we use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. We also propose a self-attention mechanism and a special regularization term for the model. As a side effect, the embedding comes with an easy way of visualizing what specific parts of the sentence are encoded into the embedding. We evaluate our model on 3 different tasks: author profiling, sentiment classification and textual entailment. Results show that our model yields a significant performance gain compared to other sentence embedding methods in all of the 3 tasks.

1 INTRODUCTION

Much progress has been made in learning semantically meaningful distributed representations of individual words, also known as word embeddings (Bengio et al., 2001; Mikolov et al., 2013). On the other hand, much remains to be done to obtain satisfying representations of phrases and sentences. Those methods generally fall into two categories. The first consists of universal sentence embeddings usually trained by unsupervised learning (Hill et al., 2016). This includes SkipThought vectors (Kiros et al., 2015), ParagraphVector (Le & Mikolov, 2014), recursive auto-encoders (Socher et al., 2011; 2013), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), etc.

The other category consists of models trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. One generally finds that specifically trained sentence embeddings perform better than generic ones, although generic ones can be used in a semi-supervised setting, exploiting large unlabeled corpora. Several models have been proposed along this line, by using recurrent networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), recursive networks (Socher et al., 2013) and convolutional networks (Kalchbrenner et al., 2014; dos Santos & Gatti, 2014; Kim, 2014) as an intermediate step in creating sentence representations to solve a wide variety of tasks including classification and ranking (Yin & Schütze, 2015; Palangi et al., 2016; Tan et al., 2016; Feng et al., 2015). A common approach in previous methods consists in creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. Additional works have also been done in exploiting linguistic structures such as parse and dependence trees to improve sentence representations (Ma et al., 2015; Mou et al., 2015b; Tai et al., 2015).

For some tasks people propose to use attention mechanism on top of the CNN or LSTM model to introduce extra source of information to guide the extraction of sentence embedding (dos Santos et al., 2016). However, for some other tasks like sentiment classification, this is not directly applicable since there is no such extra information: the model is only given one single sentence as input. In those cases, the most common way is to add a max pooling or averaging step across all time steps

[*]This work has been done during the 1st author's internship with IBM Watson.

Structured Self-Attentive Sentence Embedding

- Zhouhan et al. (2016)
- Sentence 단위를 Embedding하기 위한 기법
- Sentence 내의 관계를 파악하기 위해 Self Attention 사용
- Auto profiling, Sentiment Analysis, NLI Task에 적용

A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

Zhouhan Lin[‡◇*], **Minwei Feng**[◇], **Cicero Nogueira dos Santos**[◇], **Mo Yu**[◇],
Bing Xiang[◇], **Bowen Zhou**[◇] **& Yoshua Bengio**[‡†]
[◇]IBM Watson
[‡]Montreal Institute for Learning Algorithms (MILA), Université de Montréal
[†]CIFAR Senior Fellow
lin.zhouhan@gmail.com
{mfeng, cicerons, yum, bingxia, zhou}@us.ibm.com

ABSTRACT

This paper proposes a new model for extracting an interpretable sentence embedding by introducing self-attention. Instead of using a vector, we use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. We also propose a self-attention mechanism and a special regularization term for the model. As a side effect, the embedding comes with an easy way of visualizing what specific parts of the sentence are encoded into the embedding. We evaluate our model on 3 different tasks: author profiling, sentiment classification and textual entailment. Results show that our model yields a significant performance gain compared to other sentence embedding methods in all of the 3 tasks.

1 INTRODUCTION

Much progress has been made in learning semantically meaningful distributed representations of individual words, also known as word embeddings (Bengio et al., 2001; Mikolov et al., 2013). On the other hand, much remains to be done to obtain satisfying representations of phrases and sentences. Those methods generally fall into two categories. The first consists of universal sentence embeddings usually trained by unsupervised learning (Hill et al., 2016). This includes SkipThought vectors (Kiros et al., 2015), ParagraphVector (Le & Mikolov, 2014), recursive auto-encoders (Socher et al., 2011; 2013), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), etc.

The other category consists of models trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. One generally finds that specifically trained sentence embeddings perform better than generic ones, although generic ones can be used in a semi-supervised setting, exploiting large unlabeled corpora. Several models have been proposed along this line, by using recurrent networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), recursive networks (Socher et al., 2013) and convolutional networks (Kalchbrenner et al., 2014; dos Santos & Gatti, 2014; Kim, 2014) as an intermediate step in creating sentence representations to solve a wide variety of tasks including classification and ranking (Yin & Schütze, 2015; Palangi et al., 2016; Tan et al., 2016; Feng et al., 2015). A common approach in previous methods consists in creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. Additional works have also been done in exploiting linguistic structures such as parse and dependence trees to improve sentence representations (Ma et al., 2015; Mou et al., 2015b; Tai et al., 2015).

For some tasks people propose to use attention mechanism on top of the CNN or LSTM model to introduce extra source of information to guide the extraction of sentence embedding (dos Santos et al., 2016). However, for some other tasks like sentiment classification, this is not directly applicable since there is no such extra information: the model is only given one single sentence as input. In those cases, the most common way is to add a max pooling or averaging step across all time steps

[*]This work has been done during the 1st author's internship with IBM Watson.

Structured Self-Attentive Sentence Embedding

- Zhouhan et al. (2016)
- Sentence 단위를 Embedding하기 위한 기법
- Sentence 내의 관계를 파악하기 위해 Self Attention 사용
- Auto profiling, Sentiment Analysis, NLI Task에 적용

- Text Classification Task중 하나,
  주어진 글 작성자의 정보를 예측

A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

Zhouhan Lin[‡◇*], **Minwei Feng**[◇], **Cicero Nogueira dos Santos**[◇], **Mo Yu**[◇],
Bing Xiang[◇], **Bowen Zhou**[◇] **& Yoshua Bengio**[‡†]
[◇]IBM Watson
[‡]Montreal Institute for Learning Algorithms (MILA), Université de Montréal
[†]CIFAR Senior Fellow
lin.zhouhan@gmail.com
{mfeng, cicerons, yum, bingxia, zhou}@us.ibm.com

ABSTRACT

This paper proposes a new model for extracting an interpretable sentence embedding by introducing self-attention. Instead of using a vector, we use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. We also propose a self-attention mechanism and a special regularization term for the model. As a side effect, the embedding comes with an easy way of visualizing what specific parts of the sentence are encoded into the embedding. We evaluate our model on 3 different tasks: author profiling, sentiment classification and textual entailment. Results show that our model yields a significant performance gain compared to other sentence embedding methods in all of the 3 tasks.

1 INTRODUCTION

Much progress has been made in learning semantically meaningful distributed representations of individual words, also known as word embeddings (Bengio et al., 2001; Mikolov et al., 2013). On the other hand, much remains to be done to obtain satisfying representations of phrases and sentences. Those methods generally fall into two categories. The first consists of universal sentence embeddings usually trained by unsupervised learning (Hill et al., 2016). This includes SkipThought vectors (Kiros et al., 2015), ParagraphVector (Le & Mikolov, 2014), recursive auto-encoders (Socher et al., 2011; 2013), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), etc.

The other category consists of models trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. One generally finds that specifically trained sentence embeddings perform better than generic ones, although generic ones can be used in a semi-supervised setting, exploiting large unlabeled corpora. Several models have been proposed along this line, by using recurrent networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), recursive networks (Socher et al., 2013) and convolutional networks (Kalchbrenner et al., 2014; dos Santos & Gatti, 2014; Kim, 2014) as an intermediate step in creating sentence representations to solve a wide variety of tasks including classification and ranking (Yin & Schütze, 2015; Palangi et al., 2016; Tan et al., 2016; Feng et al., 2015). A common approach in previous methods consists in creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. Additional works have also been done in exploiting linguistic structures such as parse and dependence trees to improve sentence representations (Ma et al., 2015; Mou et al., 2015b; Tai et al., 2015).

For some tasks people propose to use attention mechanism on top of the CNN or LSTM model to introduce extra source of information to guide the extraction of sentence embedding (dos Santos et al., 2016). However, for some other tasks like sentiment classification, this is not directly applicable since there is no such extra information: the model is only given one single sentence as input. In those cases, the most common way is to add a max pooling or averaging step across all time steps

[*]This work has been done during the 1st author's internship with IBM Watson.

Structured Self-Attentive Sentence Embedding

- Zhouhan et al. (2016)
- Sentence 단위를 Embedding하기 위한 기법
- Sentence 내의 관계를 파악하기 위해 Self Attention 사용
- Auto profiling, Sentiment Analysis, NLI Task에 적용

- Text Classification Task중 하나,
  주어진 글 작성자의 정보를 예측
- 해당 논문에서는 Twitter dataset으로
  작성자의 나이를 예측

A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

Zhouhan Lin[‡◇*], **Minwei Feng**[◇], **Cicero Nogueira dos Santos**[◇], **Mo Yu**[◇],
Bing Xiang[◇], **Bowen Zhou**[◇] **& Yoshua Bengio**[‡†]
[◇]IBM Watson
[‡]Montreal Institute for Learning Algorithms (MILA), Université de Montréal
[†]CIFAR Senior Fellow
lin.zhouhan@gmail.com
{mfeng, cicerons, yum, bingxia, zhou}@us.ibm.com

ABSTRACT

This paper proposes a new model for extracting an interpretable sentence embedding by introducing self-attention. Instead of using a vector, we use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. We also propose a self-attention mechanism and a special regularization term for the model. As a side effect, the embedding comes with an easy way of visualizing what specific parts of the sentence are encoded into the embedding. We evaluate our model on 3 different tasks: author profiling, sentiment classification and textual entailment. Results show that our model yields a significant performance gain compared to other sentence embedding methods in all of the 3 tasks.

1 INTRODUCTION

Much progress has been made in learning semantically meaningful distributed representations of individual words, also known as word embeddings (Bengio et al., 2001; Mikolov et al., 2013). On the other hand, much remains to be done to obtain satisfying representations of phrases and sentences. Those methods generally fall into two categories. The first consists of universal sentence embeddings usually trained by unsupervised learning (Hill et al., 2016). This includes SkipThought vectors (Kiros et al., 2015), ParagraphVector (Le & Mikolov, 2014), recursive auto-encoders (Socher et al., 2011; 2013), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), etc.

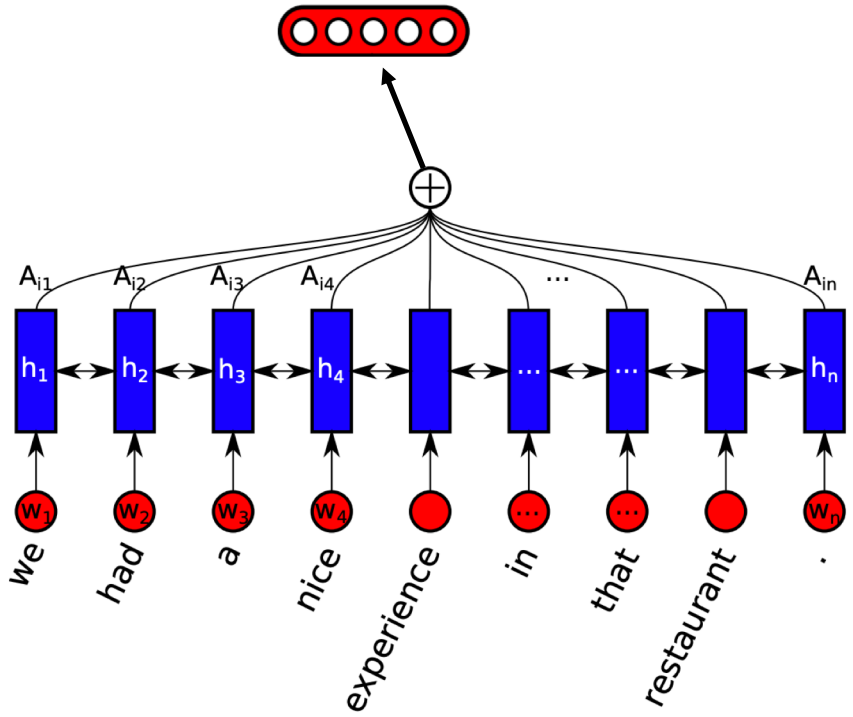
The other category consists of models trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. One generally finds that specifically trained sentence embeddings perform better than generic ones, although generic ones can be used in a semi-supervised setting, exploiting large unlabeled corpora. Several models have been proposed along this line, by using recurrent networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), recursive networks (Socher et al., 2013) and convolutional networks (Kalchbrenner et al., 2014; dos Santos & Gatti, 2014; Kim, 2014) as an intermediate step in creating sentence representations to solve a wide variety of tasks including classification and ranking (Yin & Schütze, 2015; Palangi et al., 2016; Tan et al., 2016; Feng et al., 2015). A common approach in previous methods consists in creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. Additional works have also been done in exploiting linguistic structures such as parse and dependence trees to improve sentence representations (Ma et al., 2015; Mou et al., 2015b; Tai et al., 2015).

For some tasks people propose to use attention mechanism on top of the CNN or LSTM model to introduce extra source of information to guide the extraction of sentence embedding (dos Santos et al., 2016). However, for some other tasks like sentiment classification, this is not directly applicable since there is no such extra information: the model is only given one single sentence as input. In those cases, the most common way is to add a max pooling or averaging step across all time steps

[*]This work has been done during the 1st author's internship with IBM Watson.

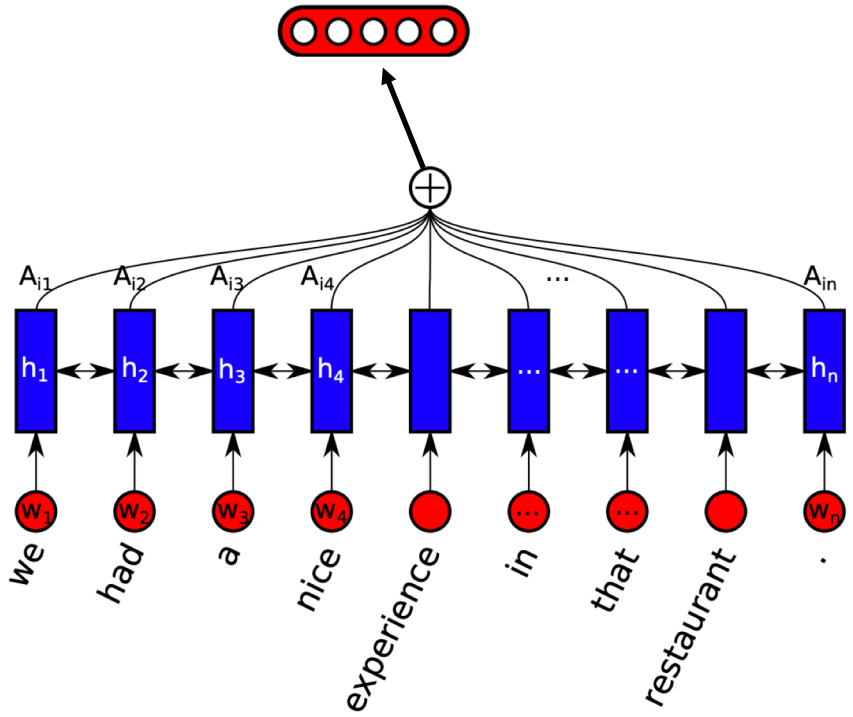
Structured Self-Attentive Sentence Embedding

Structured Self-Attentive Sentence Embedding



Structured Self-Attentive Sentence Embedding

$Input$
$S = (w_1, w_2, \ldots, w_n)$



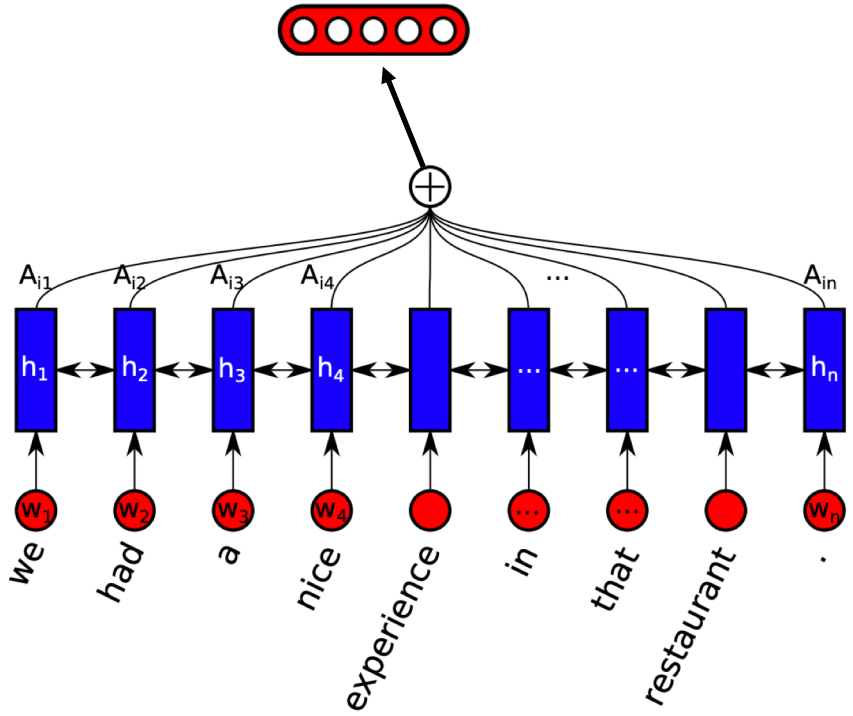
Structured Self-Attentive Sentence Embedding

Input

$S = (w_1, w_2, \ldots, w_n)$

$\vec{h} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}})$

$\overleftarrow{h} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}})$



Structured Self-Attentive Sentence Embedding

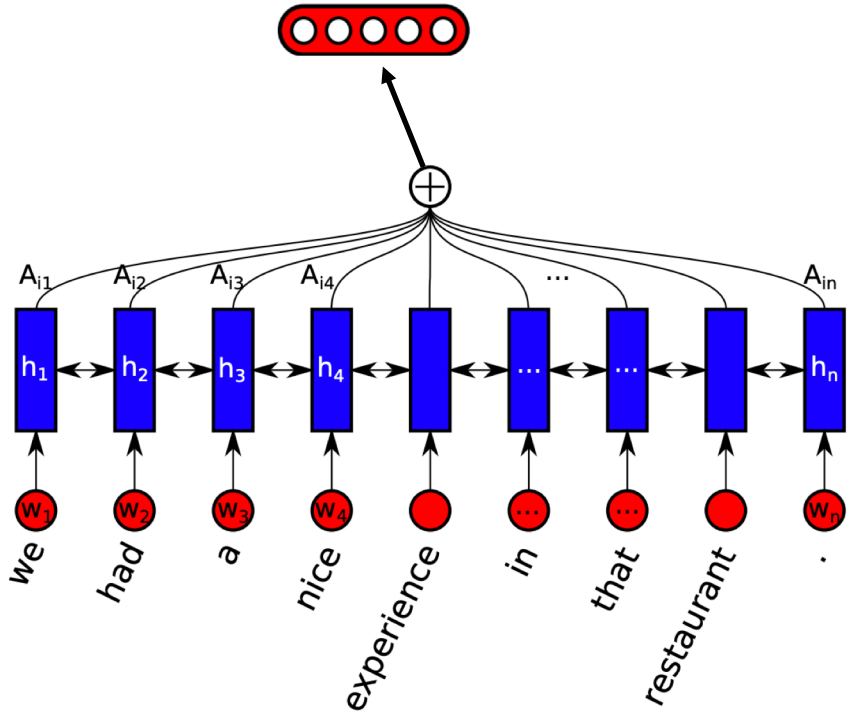
Input

$S = (w_1, w_2, \dots, w_n)$

$\vec{h} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}})$

$\overleftarrow{h} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}})$

$H = (h_1, h_2, \dots, h_n)$



Structured Self-Attentive Sentence Embedding

Input

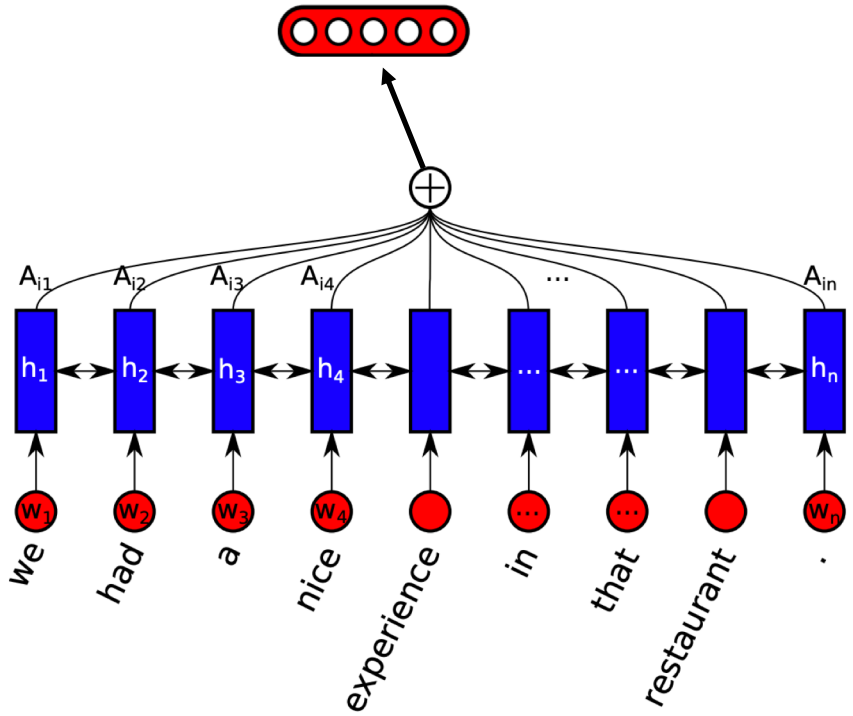
$S = (w_1, w_2, \dots, w_n)$

$\vec{h} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}})$

$\overleftarrow{h} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}})$

$H = (h_1, h_2, \dots, h_n)$

$a = softmax(w_{s2} \tanh(W_{s1} H^T))$



Structured Self-Attentive Sentence Embedding

$Input$

$S = (w_1, w_2, \dots, w_n)$

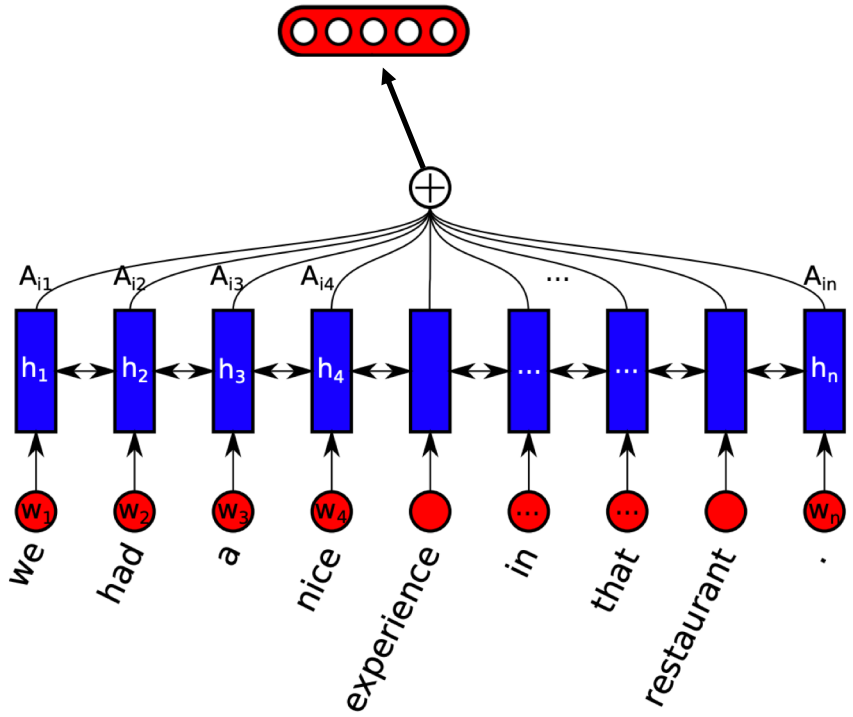
$\vec{h} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}})$

$\overleftarrow{h} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}})$

$H = (h_1, h_2, \dots, h_n)$

$a = softmax(w_{s2} \tanh(W_{s1} H^T))$

Self Attention



Structured Self-Attentive Sentence Embedding

$Input$

$S = (w_1, w_2, \dots, w_n)$

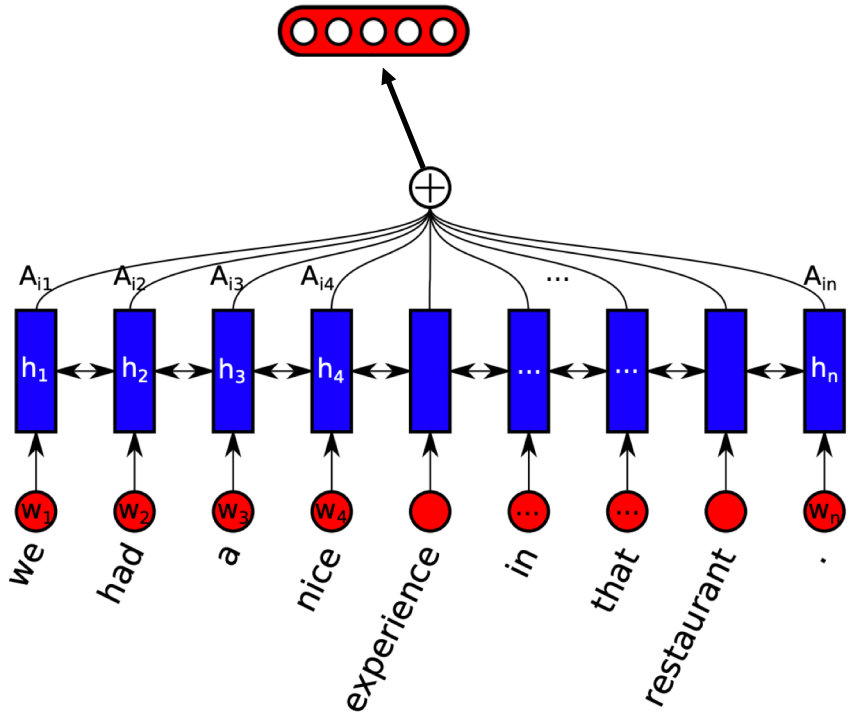
$\vec{h} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}})$

$\overleftarrow{h} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}})$

$H = (h_1, h_2, \dots, h_n)$

$a = softmax(w_{s2} \tanh(W_{s1} H^T))$

$output = a \odot H$



Another methods to enrich NMT performance

Another methods to enrich NMT performance

Another methods to enrich NMT performance

- **Teacher Forcing**
- **Beam Search**

Another methods to enrich NMT performance

Teacher Forcing

Another methods to enrich NMT performance

Beam Search