Graph-based Dependency Parsing with Neural Network

남궁영
한국해양대학교 컴퓨터공학과
young_ng@kmou.ac.kr

2019. 04. 26.

국립 한국해양대학교
KOREA MARITIME AND OCEAN UNIVERSITY

Progress in Graph-based DP with NN

- **Pei et al.**
  An Effective Neural Network Model for Graph-based Dependency Parsing
  (ACL 2015, pp. 313-322)

Bahdanau's attention

- **Kiperwasser and Goldberg**
  Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations
  (ACL 2016, vol. 4, pp. 313-327)

- **Cheng et al.**
  Bi-directional Attention with Agreement for Dependency Parsing
  (ACL 2016)

Luong's attention

- **Hashimoto et al.**
  A Joint Mani-task Model: Growing a Neural Network for Multiple NLP Tasks
  (ICLR 2017)

- **Dozat and Manning**
  Deep Biaffine Attention for Neural Dependency Parsing
  (ICLR 2017)

Machine Learning for Dependency Parsing

Graph-based Model

❖ Use ML **to assign a weight or probability** to each possible edge

❖ Construct a maximum spanning tree(MST) from these weighted edges

Transition-based Model

❖ Use ML for predicting the operations of the transition algorithm

❖ Can train any multi-class machine learning classifier on features extracted from the stack, buffer, and previous arc actions in order **to predict the next action**

| Dozat and Manning. Deep Biaffine Attention for Neural Dependency Parsing. ICLR 2017

I Graph-based Parser

❖ **Arc-factored** graph based approach (McDonald et al. 2005)

- **Decompose** the score of a tree
  to the sum of the score of its head-modifier arcs (h, m)

$$parse(s) = \sum_{(h,m) \in y} score\big(\phi(s, h, m)\big)$$

local

global

scoring function

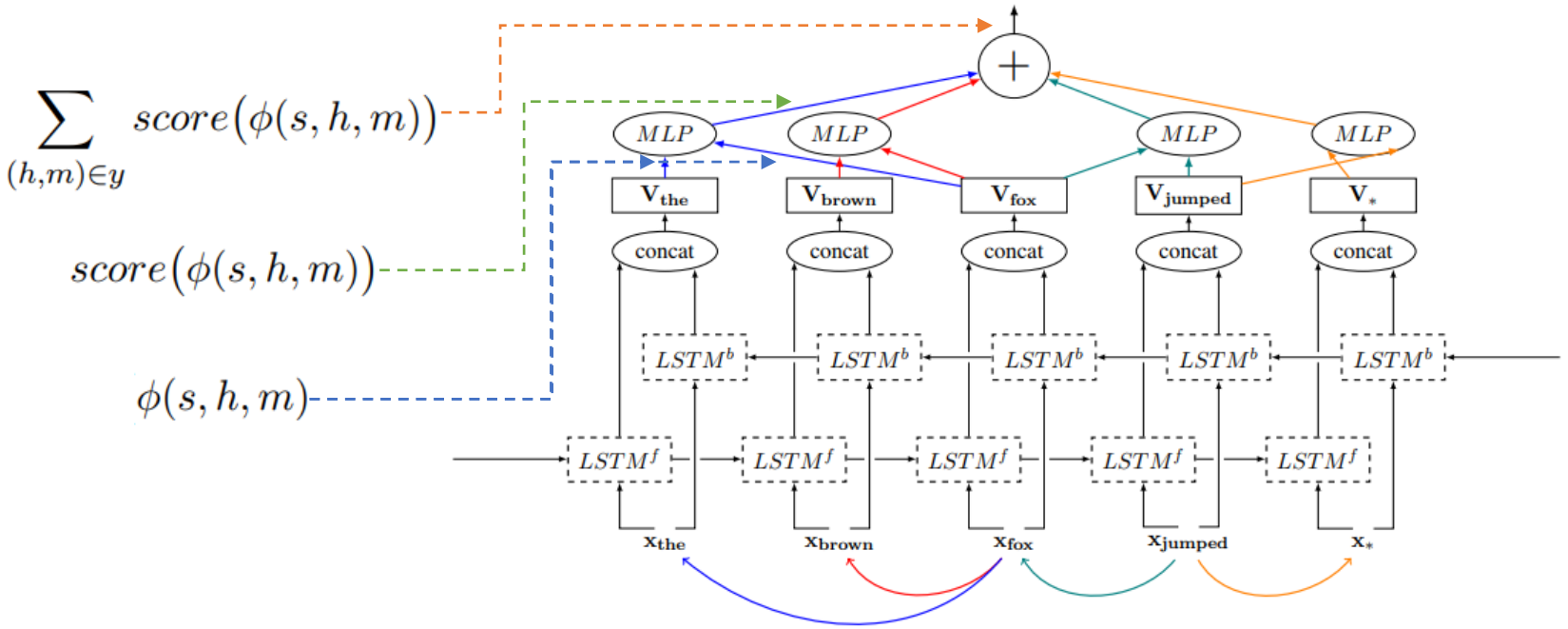
- **Linear model**
- **MLP**

- **Collin's**
- **Eisner's**
- **Chu-Liu-Edmond's (MST)**

| Kiperwasser and Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. ACL. vol. 4, pp. 313-327)
| R. McDonald et al. 2005. Online large-margin training of dependency parsers. ACL. pp. 91-98

I Kiperwasser & Goldberg (2016)

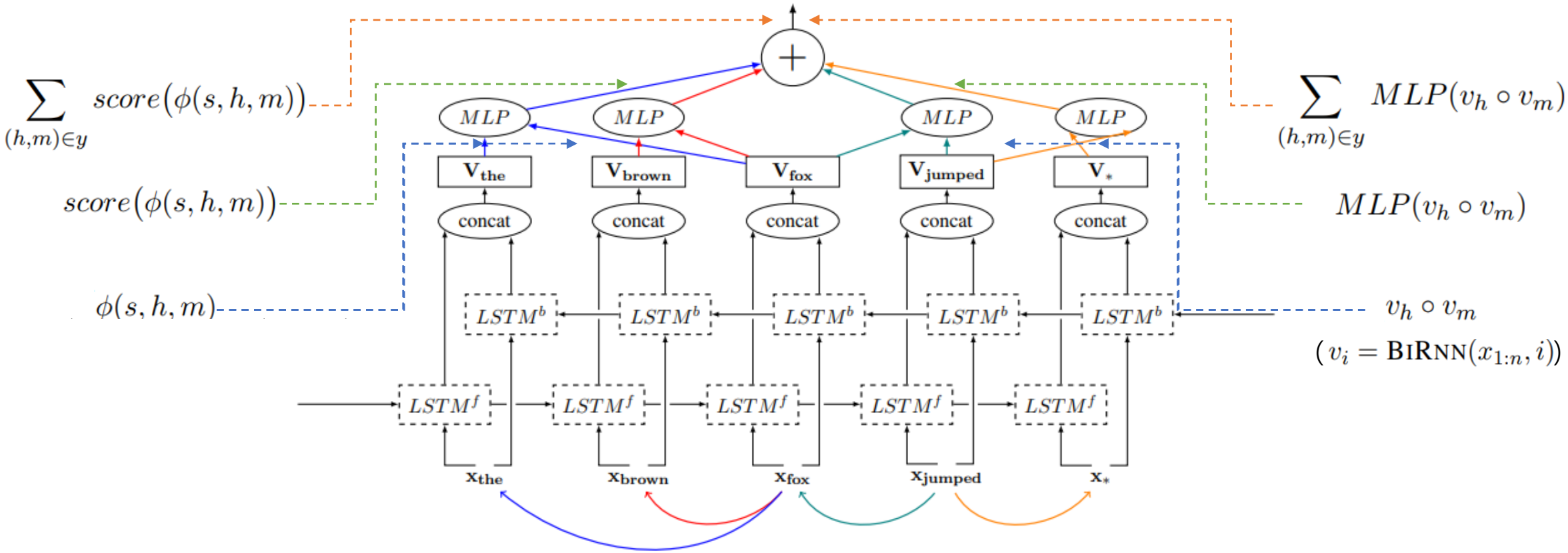
- Uses the same kind of **attention mechanism** as Bahdanau et al. (2014)
- LSTM's recurrent output vector for **each word** + each **possible head**'s recurrent vector
- The result is used as **input** to an MLP ← **scores** each dependency arc
- Individual arc scores are **summed** to produce the final score
- The predicted tree structure : each word depends on **highest-scoring head**



| Kiperwasser and Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. ACL. vol. 4, pp. 313-327)

I Kiperwasser & Goldberg (2016)

❖ The neural model scheme of the graph-based parser when calculating the score of a given parse tree

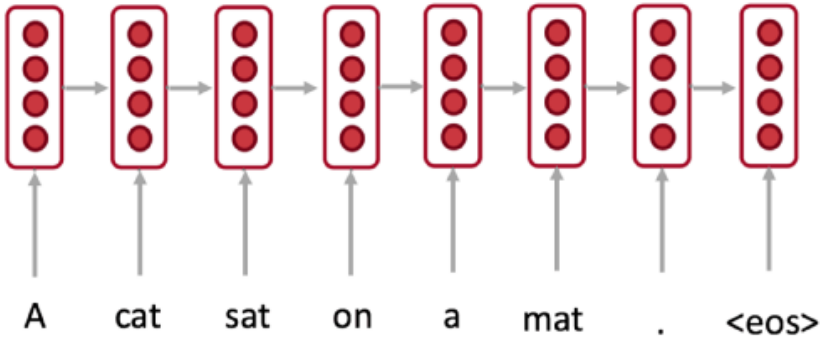


$$parse(s) = \arg\max_{y \in \mathcal{Y}(s)} \sum_{(h,m)\in y} MLP(v_h \circ v_m)$$

| Kiperwasser and Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. ACL. vol. 4, pp. 313-327)

II Attention mechanism
for NMT

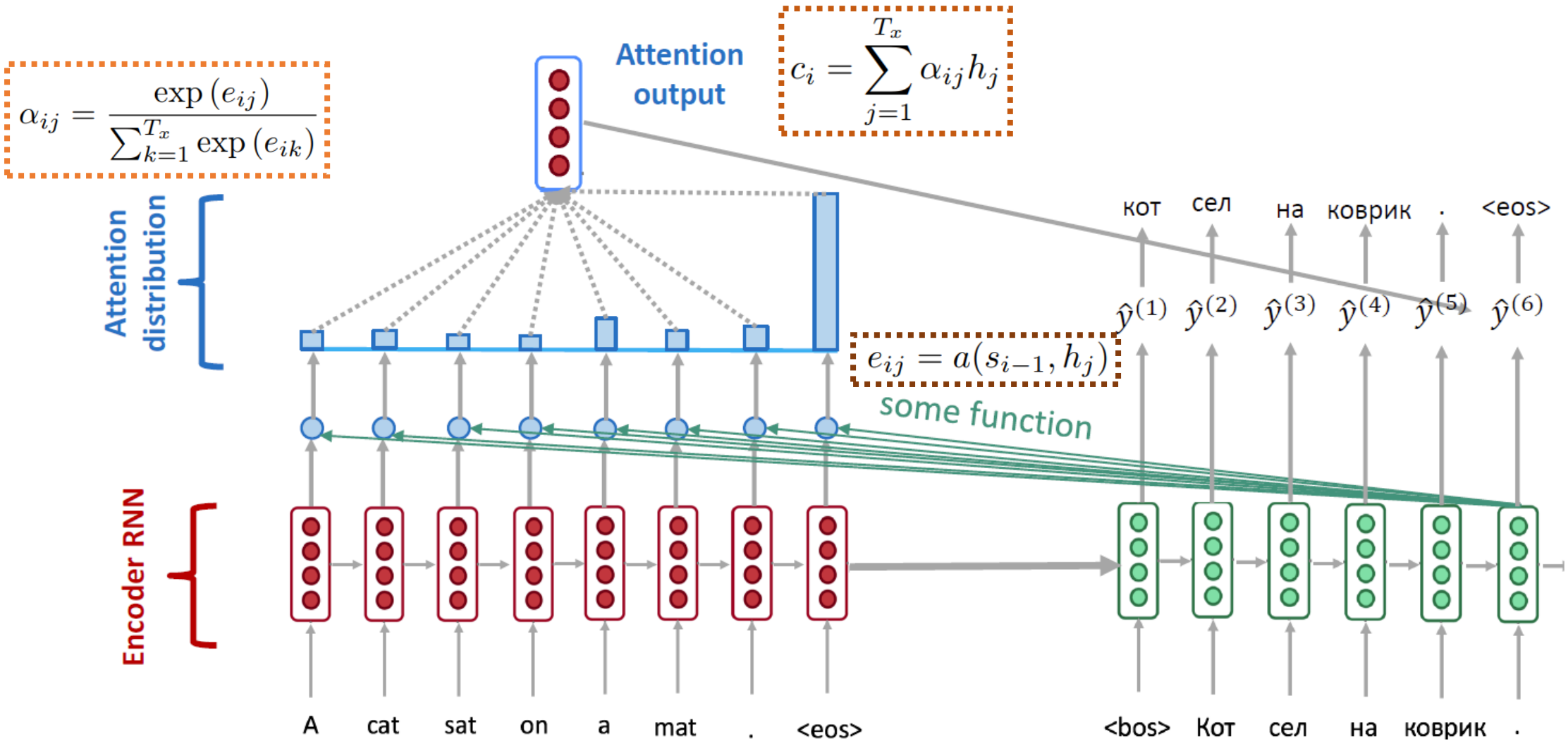
1. **Compare** target and source hidden states
2. Convert into **alignment weights**
3. Build **context vector**: weighted sum
4. Compute the **next hidden state**
5. Predict the **next word**



| https://github.com/yandexdataschool/nlp_course/tree/master/week04_seq2seq

| https://cs224d.stanford.edu/lectures/CS224d-Lecture15.pdf

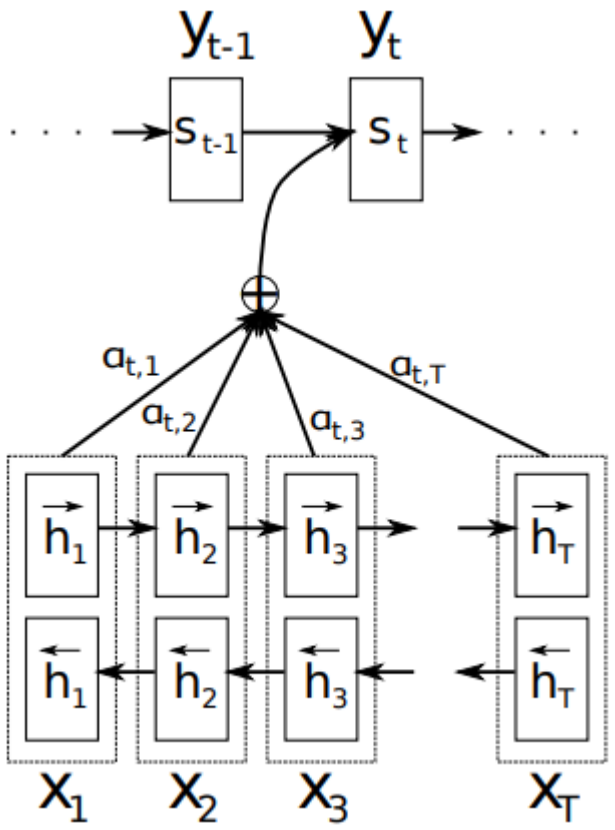
Attention mechanism
for NMT



| Elena Voita. 2018. Seq2seq and Attention. p. 80. (Yandex Research, Univ. of Amsterdam.)

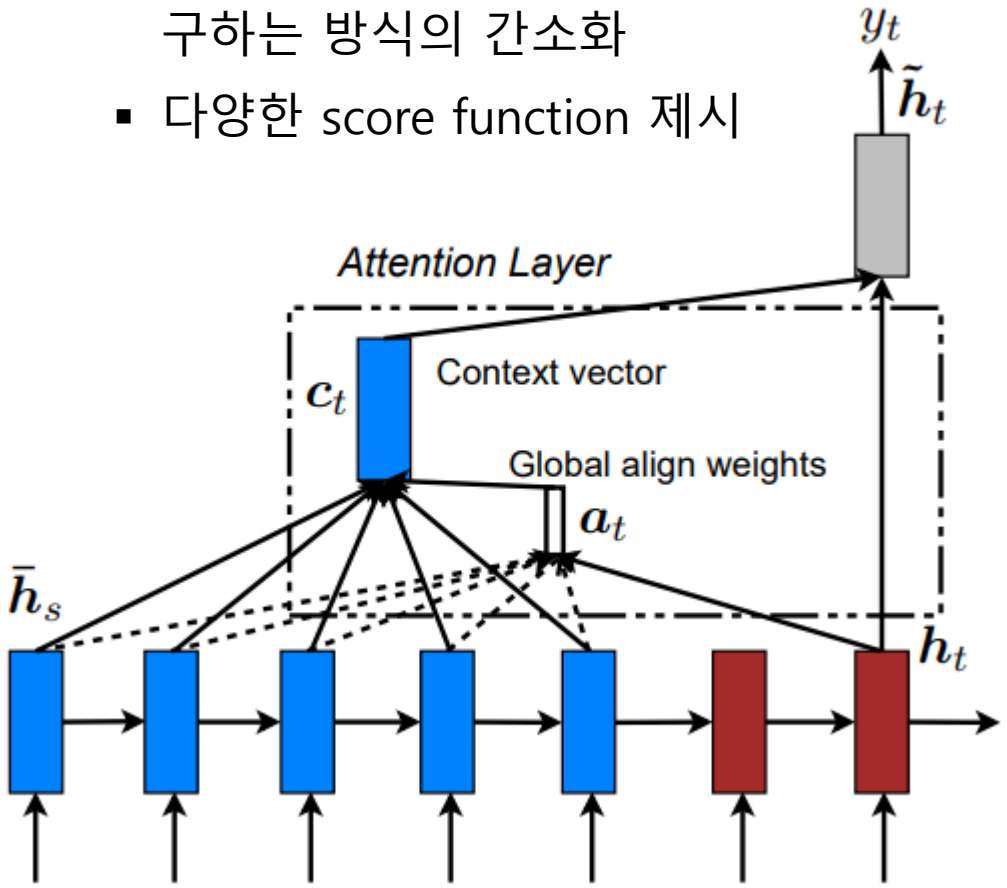
Bahdanau vs. Luong

❖ Bahdanau's attention



❖ Luong's attention

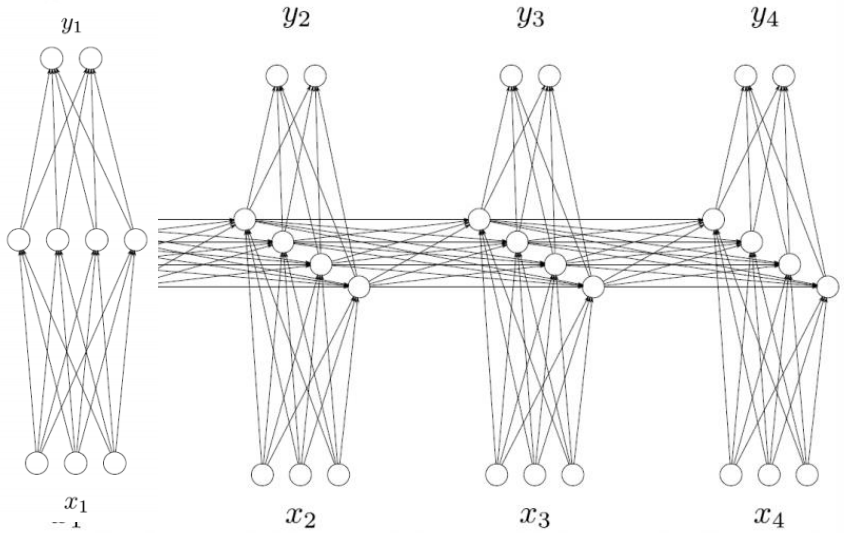
- Decoder의 Hidden state vector 구하는 방식의 간소화
- 다양한 score function 제시



| D. Bahdanau et al. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473

| T. Luong et al. 2015. Effective Approaches to Attention-based Neural Machine Translation. arXiv: 1508.04025

Terminology



❖ **Memory**

- Neural Network + Memory → Recurrent Neural Network

❖ **Attention**

❖ **Positional encoding**

- transformer: RNN (X), CNN (X)
  - 나는 학교에 간다 vs. 간다 나는 학교에
- 단어의 sequence order를 이용하기 위해 단어의 position 정보 추가 필요
- word embedding + positional encoding vector
- 절대적

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

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

- 상대적
  - P. Shaw et al. Self-Attention with Relative Position Representations (2018)

감 사 합 니 다.

남궁영
한국해양대학교 컴퓨터공학과
young_ng@kmou.ac.kr