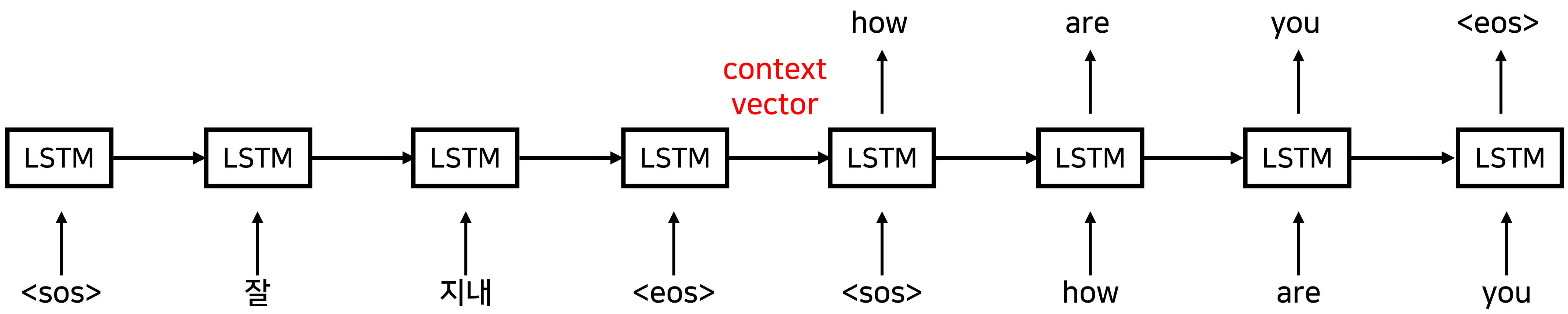
Neural Machine Translation by Jointly Learning to Align and Translate

Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio (ICLR 2015)

단국대학교 모바일시스템공학과 양윤성

Contents

1-(1). Review Seq2Seq



Seq2Seq Model

- Encoder-decoder structure of multi-layer LSTM
- Using context vector of fixed length

1-(1). Review Seq2Seq

RNN Encoder

- Map the input sentence $x = (x_1, \ldots, x_{Tx})$ into a fixed-length context vector c
- Function f() and q() are non-linear function (ex. LSTM)

$$h_t = f\left(x_t, h_{t-1}\right)$$

$$c = q\left(\{h_1, \cdots, h_{T_x}\}\right)$$

1-(1). Review Seq2Seq

RNN Decoder

- Predict the output sentence $y = (y_1, \ldots, y_{Ty})$ by maximizing the probability
- St is the hidden state of the RNN

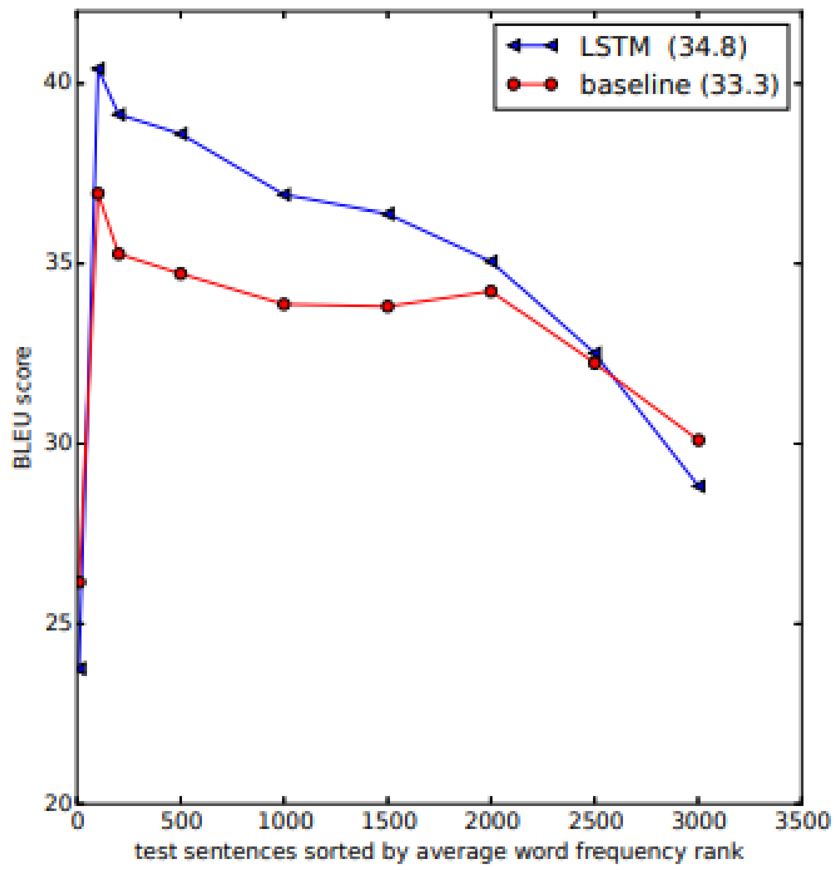
$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c)$$

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

1-(2). Limit of Seq2Seq

Bottleneck Problem?

- Encoder compresses any sentence to fixed size
- Long input sequence → High probability of information loss
- Significant impact on the performance of encoder-decoder model



1-(2). Limit of Seq2Seq

Problems mentioned in the paper

- Less performance even with LSTM
- An awkward later translation

➢ **Combining and utilizing all hidden states!**

- Searching set of positions in source sentence where the most relevant information is concentrated
- Learning model to align and translate jointly

Contents

2-(1). Embedding

- Vectorize the meaning of a word (Input of encoder & decoder)

- Limit of One Hot Encoding
  - 1 is only one, so many 0 → Inefficient & waste of memory
  - Unable to express similarities or characteristics

Word	Index	One-Hot Vector
I	0	[1, 0, 0, 0, ···, 0]
You	1	[0, 1, 0, 0, ···, 0]
We	3	[0, 0, 1, 0, ···, 0]
···	···	···
World	502	[0, 0, ···, 1, ···, 0]

2-(1). Embedding

Word2Vec

Ex) "hello my friend"

- Window size = 1

Word	Neighbor
hello	my
my	hello
my	friend
friend	my

- Window size = 2

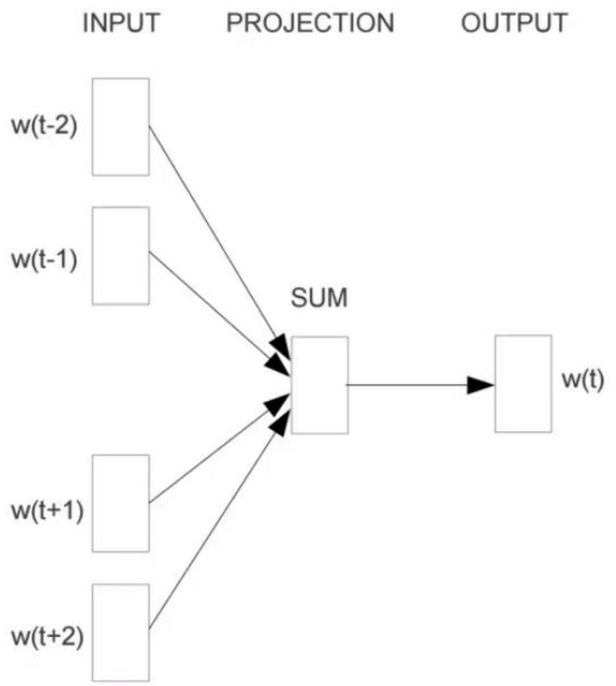
Word	Neighbor
hello	my
hello	friend
my	hello
my	friend
friend	my
friend	hello

2-(2). Word2Vec

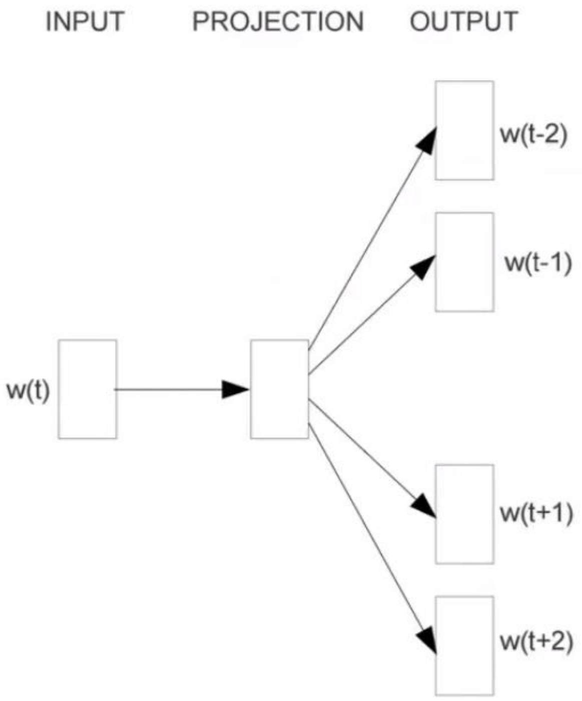
CBOW : Predict center word from context words

Skip-Gram : Predict context words from center word

● CBOW

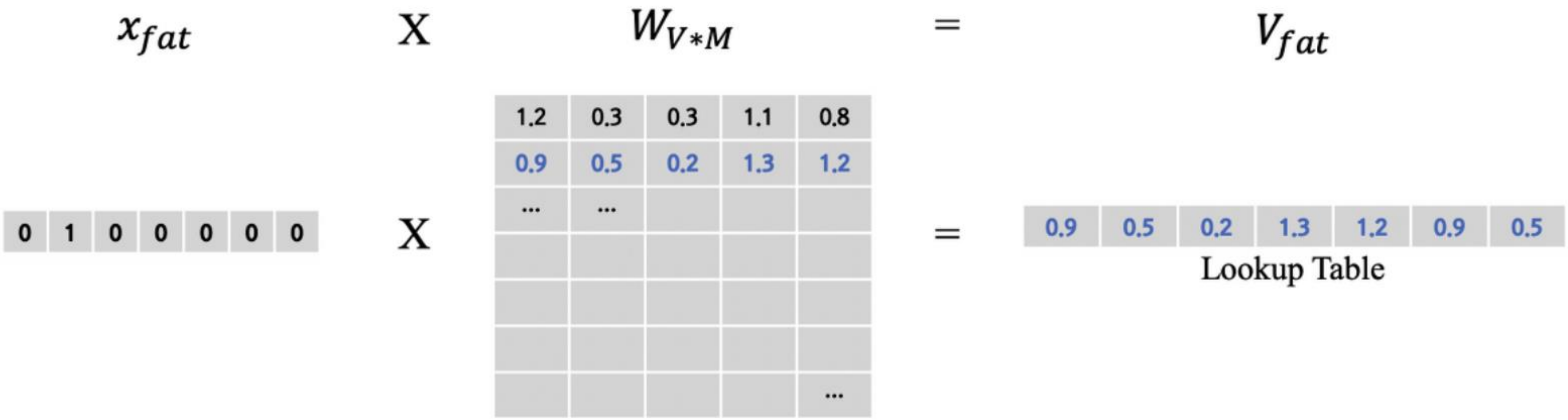


● Skip-Gram



2-(2). Word2Vec

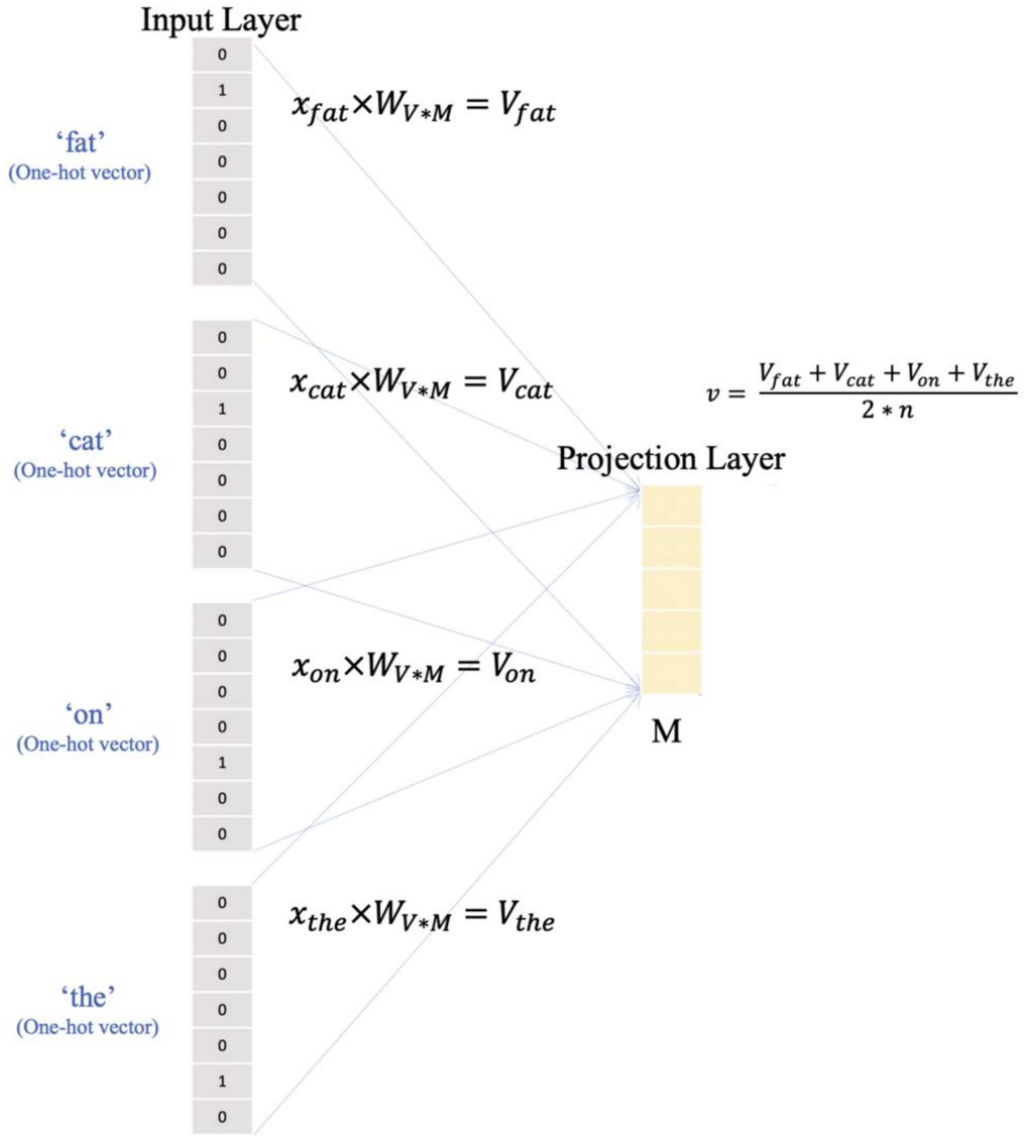
1. **Convert context words to one-hot vectors(1*V)**
2. **Multiply W(V*M) → Each embedding vector(1*M)**
   - V : The number of words, M : Embedded dimensions
   - W initially contains random values



2-(2). Word2Vec

3. Calculate avg. of vectors

→ Projection(hidden) layer(1*M)

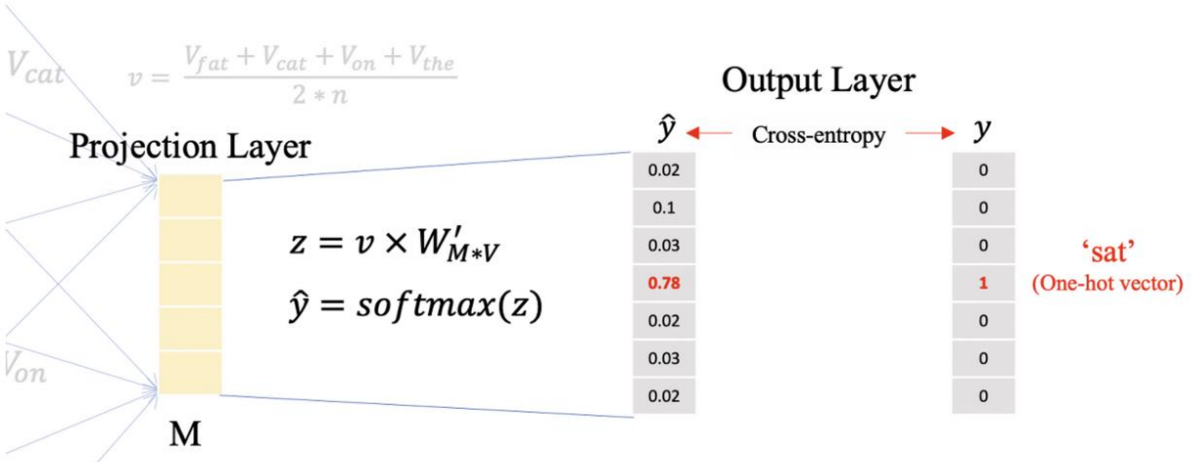


2-(2). Word2Vec

4. Calculate $\hat{y}$ and compare y (center word's one-hot vector)

5. Calculate difference between two vectors

(Use cross-entropy as loss function)



$$cost(\hat{y}, y) = -\sum_{j=1}^{V} y_j log(\hat{y})$$

$$Q(X = c_1) = 0.2 \qquad P(X = c_1) = 0$$
$$Q(X = c_2) = 0.7 \qquad P(X = c_2) = 1$$
$$Q(X = c_3) = 0.1 \qquad P(X = c_3) = 0$$

$$\begin{aligned} H(P,Q) &= -\sum P(x) \log Q(x) \\ &= -(0 \cdot \log 0.2 + 1 \cdot \log 0.7 + 0 \cdot \log 0.1) \\ &= -\log 0.7 \approx 0.357 \end{aligned}$$

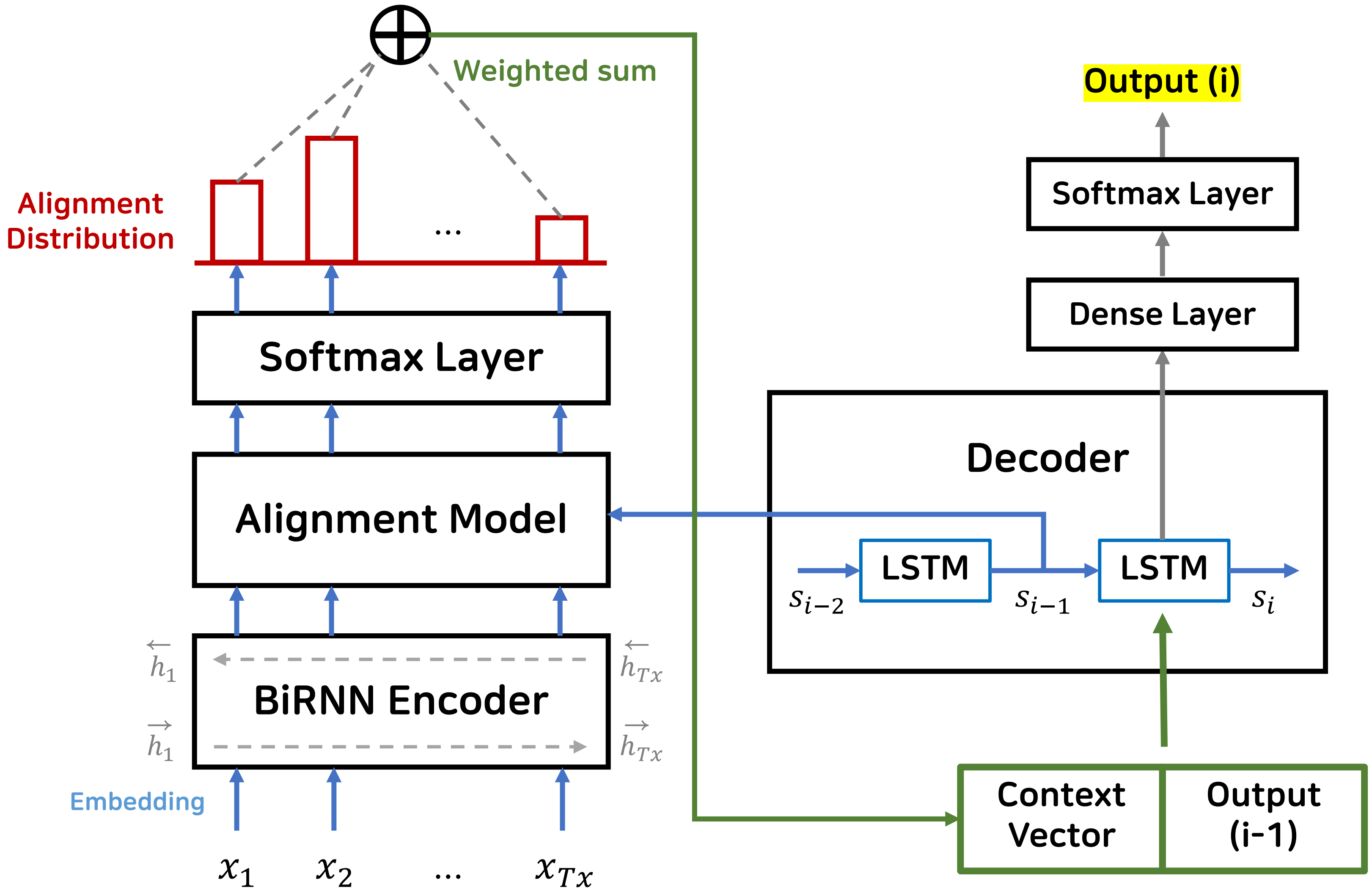
2-(2). Word2Vec

6. Using backpropagation to learn the weight matrix

- Use argmin function to select less difference
- Update all embedded vectors
- Θ : Set of all weights

$$\hat{\theta} = \arg\min_{\theta} cost(\hat{y}, y)$$

2-(3). Whole Process



2-(4). BiRNN Encoder

Bidirectional RNN

- Forward RNN : Reads input sequence from beginning to end

$$(\overrightarrow{h}_1, \cdots, \overrightarrow{h}_{T_x})$$

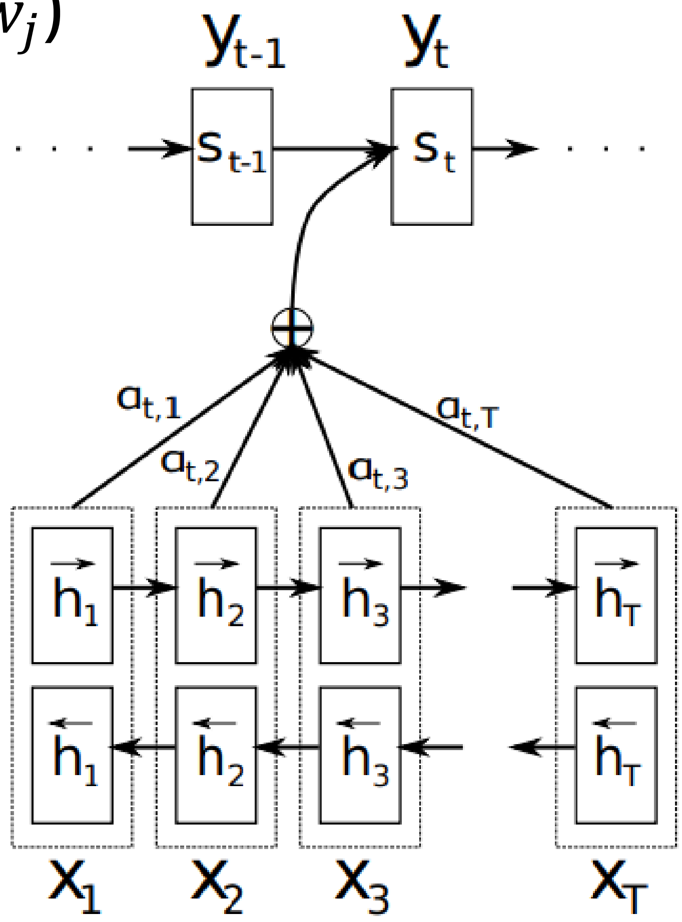
- Backward RNN : Reads input sequence from end to beginning

$$(\overleftarrow{h}_1, \cdots, \overleftarrow{h}_{T_x})$$

2-(4). BiRNN Encoder

- **Annotation** (h_j) : New hidden state of each word (w_j)
  - Something representation
  - Contain the summaries of both the preceding words and the following words

$$h_j = \left[\overrightarrow{h}_j^\top ; \overleftarrow{h}_j^\top\right]^\top$$



2-(5). Compute Alignment

- **Hidden state in decoder (s_i)** : Information in time i
- **Energy (e_{ij})** : Calculate s_{i-1} and h_j to use alignment model

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

- **Alignment model**

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

  - V_a, W_a, U_a : Parameter matrix used to incorporate encoder/decoder's hidden state into the weights

2-(5). Compute Alignment

- **Weight (α_{ij})** : Probability that target word y_i is aligned to source word x_j
  - Function exp : $\exp(\boldsymbol{x}) = \boldsymbol{e}^{\boldsymbol{x}}$
  - Normalize to softmax layer → Final probability

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

2-(6). Make Context Vector

- **Weighted Sum (c_i)**
  - Sum (j-th weight * j-th annotation) of j = 1 to j = T_x
  - Final result → i-th context vector

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

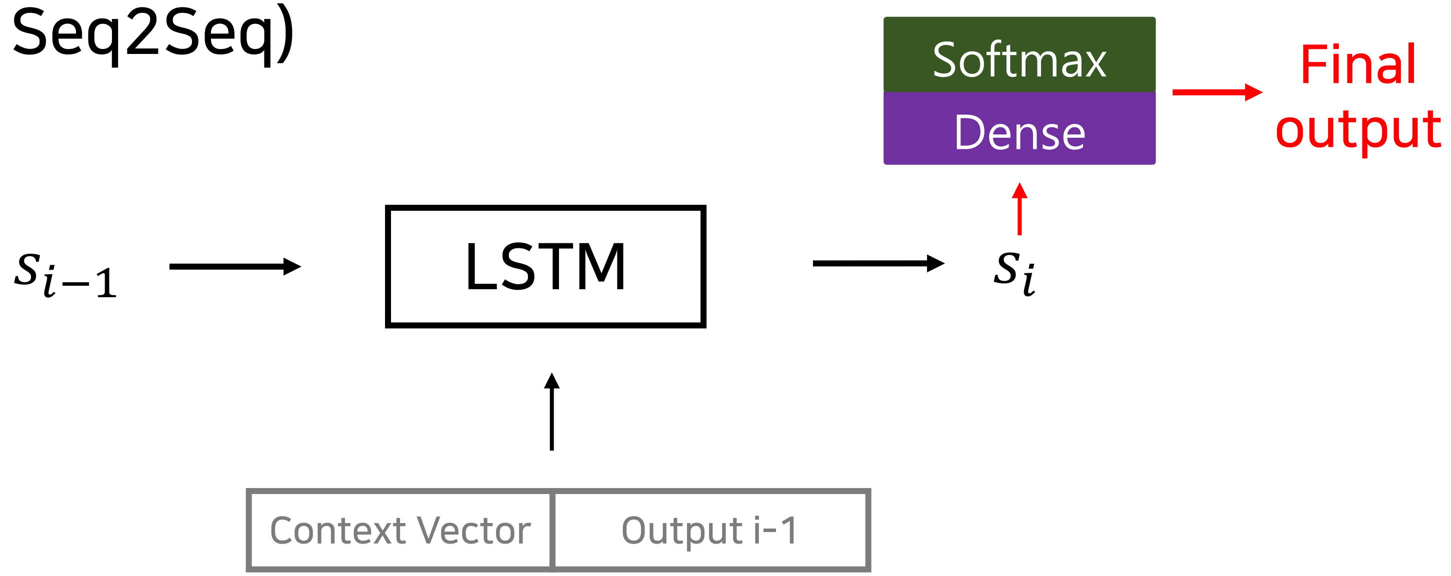
2-(7). Decoder

- Operate i-th context vector, pre word, and pre hidden state into LSTM

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

- Predict the final word through dense layer and softmax layer

  (Same as before Seq2Seq)



Contents

3-(1). Settings

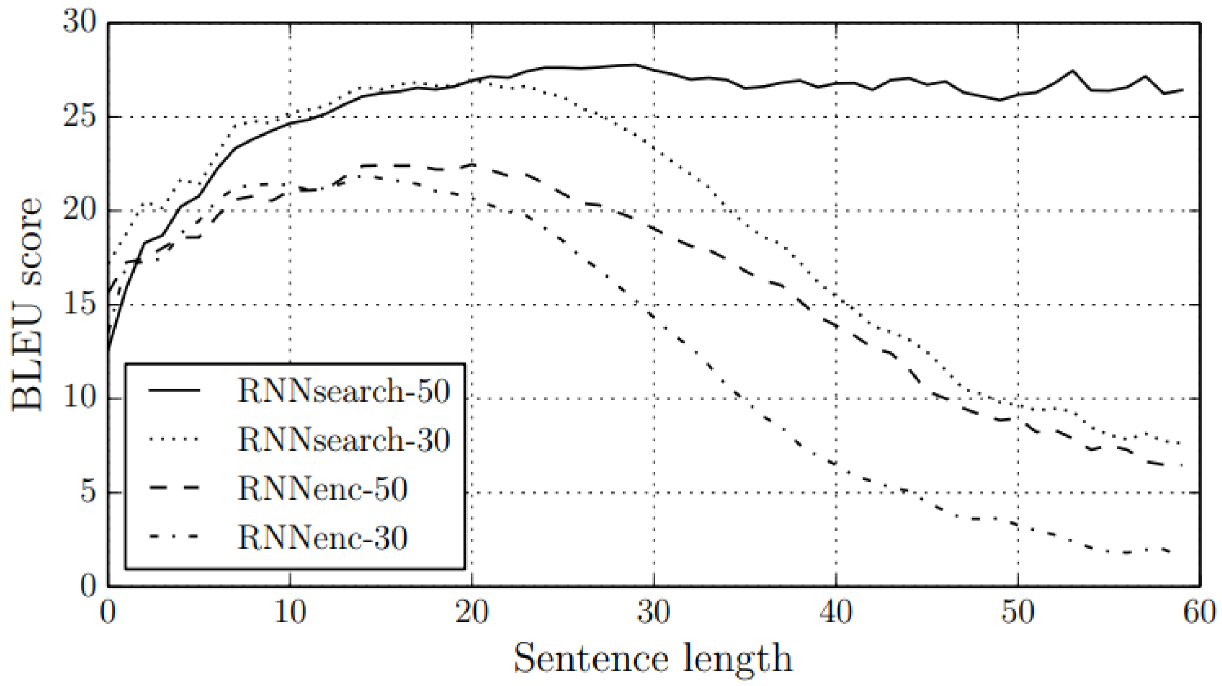
- WMT'14 English-French parallel corpus dataset
- RNN Encoder-Decoder model (RNNencdec-30, RNNencdec-50)
- RNN Search model (RNNsearch-30, RNNsearch-50)
- RNNsearch-50* : Trained much longer
- Moses : Best open source SMT → for comparison
- Using BLEU score

3-(2). Result – BLEU score

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

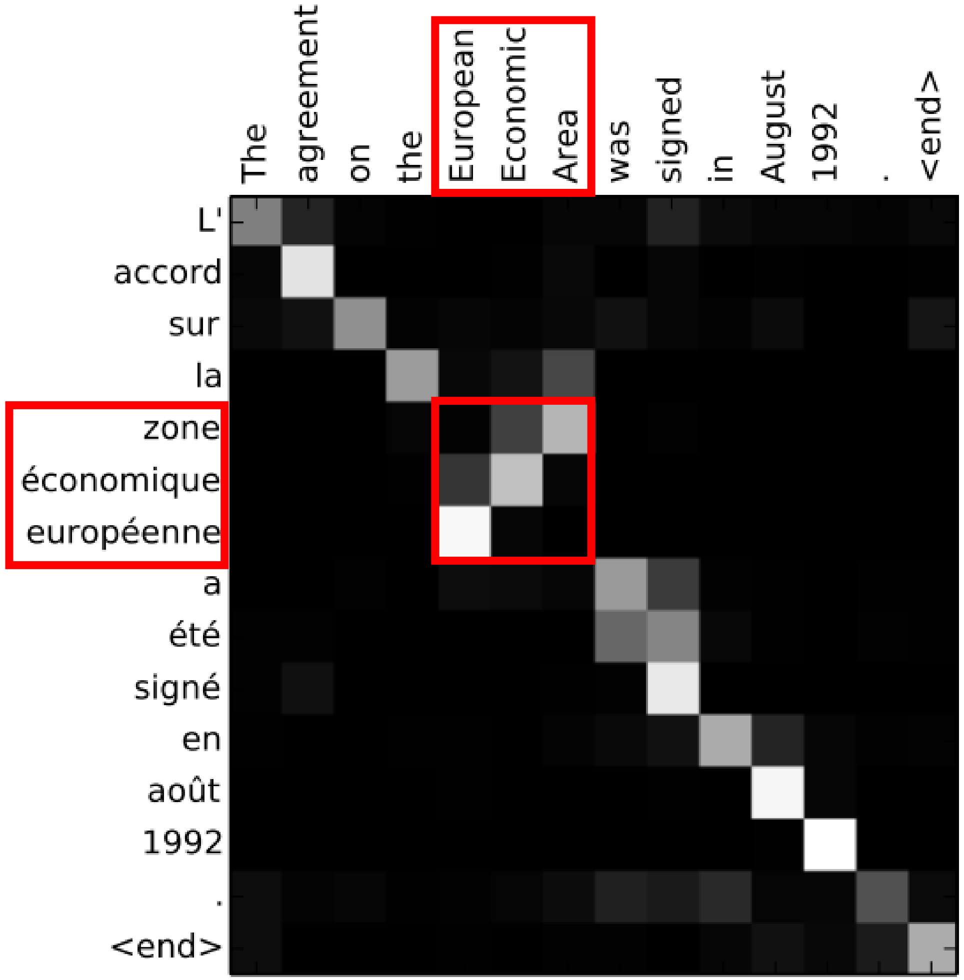
- No UNK : Disallowed the models to generate unknown(UNK) token
- UNK token : Using UNK token instead of words it doesn't know
- High quality than Moses when No UNK

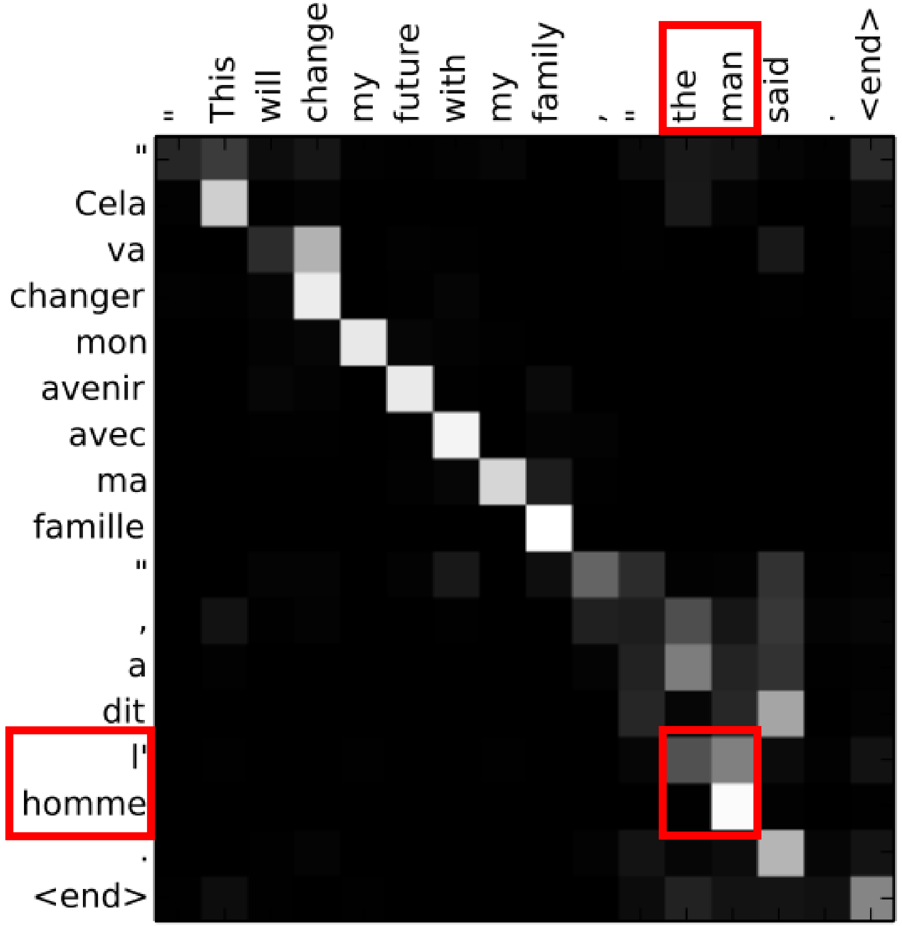
3-(2). Result – BLEU score



- Before model : Reduced performance in changing to fixed length vector
- After model : Performance doesn’t decrease even with longer sentences

3-(3). Result – Alignment





3-(3). Result – Alignment

- X-axis : Input sequence (English)
- Y-axis : Output sequence (French)
- Grayscale Annotation (0 : Black, 1 : White)
- The larger annotation value → The more relevant it is
- An accurate prediction even if the word order is opposite

Contents

4. Conclusion

- Out of vector of fixed length → Explore annotation for each word generation
- Focus only on information relevant to the generation of the next target word
  - More accurate translation results
- Achieve performance comparable to the existing phrase-based SMT
- Finding better handle unknown or rare words is required

Thank You!

Neural Machine Translation by Jointly Learning to Align and Translate