Neural machine translation by jointly learning to align and translate

Name
김용
자연어 처리
2024/04/02

TRAIN AND TEST

3. Learning To Align and Translate

Input sentence

$$\boldsymbol{x} = (x_1, x_2, x_3, \ldots, x_{T_x})$$

Context vector

$$\mathbf{c}$$

Common approach of RNN

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

TRAIN AND TEST

3. Learning To Align and Translate

$$c = q(\{h_1, h_2, \ldots, h_{T_x}\})$$

$h_t \in R^n$ hidden state at time t,

c is a vector generated from the sequence of hidden states.

f -> LSTM

$q(\{h_1, h_2, \ldots, h_T\}) = h_T$ -> seq2seq에 따르면 마지막 input token의 hidden state가 context vector로 쓰이기 때문에

TRAIN AND TEST

3. Learning To Align and Translate

Decoding 방식

$$p(y) = \prod_{t=1}^{T} p(y_t \mid \{y_1, y_2, \ldots, y_{t-1}\}, c)$$

이때 $y = \left(y_1, \ldots., y_{T_y}\right)$.

$$p(y_t \mid \{y_1, y_2, \ldots, y_{t-1}\}, c) = \mathrm{g}(y_{t-1}, s_t, c)$$

g는 y_t를 return하는 확률을 output으로 하고
s_t는 hidden state of the RNN

TRAIN AND TEST

3. Learning To Align and Translate

$$p(y_i \mid \{y_1, y_2, \ldots, y_{i-1}\}, c) = \mathrm{g}(y_{i-1}, s_i, c_i)$$

전 슬라이드의 식과 다른 점이 바로 'c'

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

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

TRAIN AND TEST

3. Learning To Align and Translate

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

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

TRAIN AND TEST

3. Learning To Align and Translate

3.2 Encoder: Bidirectional RNN for annotating sequences

Want annotation of each word to summarize not only the preceding words, but also the following words.
➔ Bidirectional RNN

$$h_j = [\vec{h}_j^T, \overleftarrow{h}_j^T]$$

TRAIN AND TEST

4. Experiment settings

- 이와 같은 과정을 영어-불어 번역 task에 사용하여 모델을 평가

- WMT’ 14의 Bilingual parallel corpora 사용

4. Experiment settings

4.1 Dataset

- WMT’ 14는 English-French parallel corpora.
- 대략 800M words가 넘는데 그 중 345M words만 사용
- Validation set으로 news-test-2012와 news–test-2013사용 / test set으로는 news-test-2014를 사용
- 토큰화에서는 가장 많이 쓰인 30000개의 단어를 쓰고, 나머지는 [UNK]토큰으로 처리

TRAIN AND TEST

4. Experiment settings

4.2 Models

- 2가지 모델 사용 (일반 RNN Encoder-Decoder, 논문의 모델)

- 1000 hidden units(Encoder, Decoder) / multilayer network with a single maxout hidden layer
  - Affine + 최댓값 두개의 층으로 구성, -> np.maximum(0, np.dot(W, x) + b)

- Minibatch stochastic gradient descent algorithm

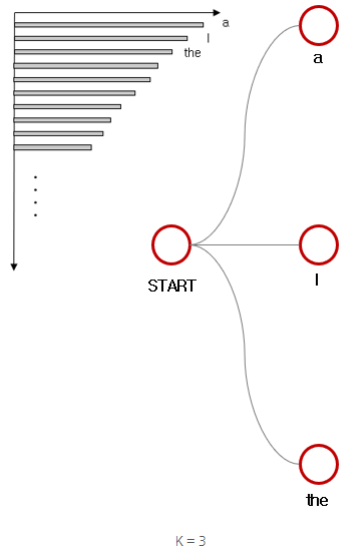
- Beam search 사용(greedy search와 exhaustive search 중간)
  - Beam serach에서 k를 지정. 특정 시점마다 가장 확률이 높은 단어 k개를 선택하고 다음 시점에서 k^2개중에서 k개를 선택한다

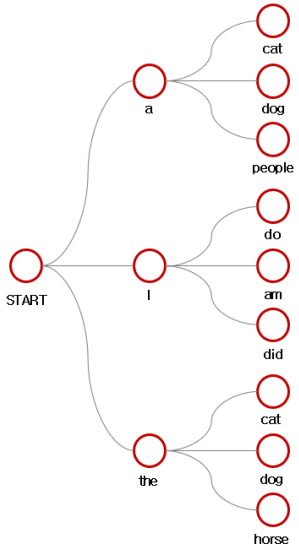
TRAIN AND TEST

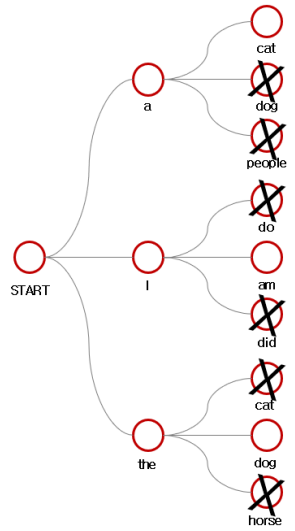
4. Experiment settings

4.2 Models

- Beam search







TRAIN AND TEST

5. Results

5.1 Quantitative Results

- Translation의 결과를 BLEU score로 측정
- RNNsearch > RNNencdec
- Robust to the length of sentences. : RNNserach-30 > RNNencdec-50

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

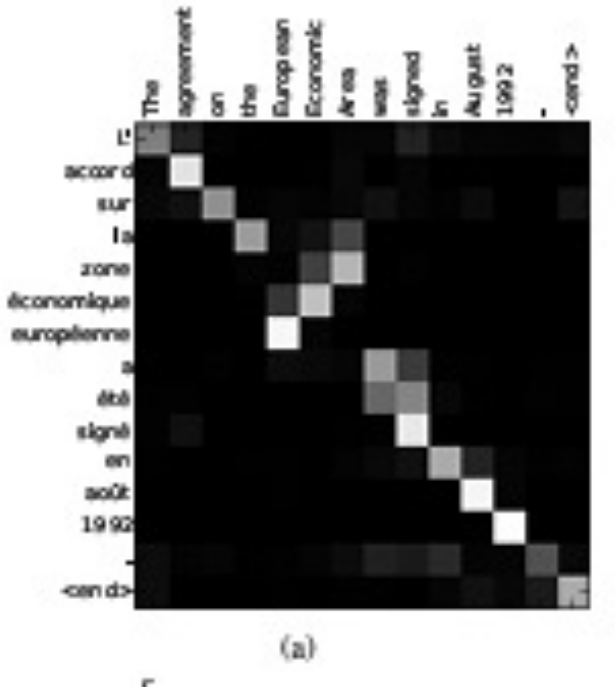
TRAIN AND TEST

5. Results

5.2 Quantative analysis

- 오른쪽 그림에서 볼 수 있듯이 source 문장의 어떤 단어가 target문장의 어떤 단어와 매치되는지

- 보통 diagonal하지만 다른 경우도 있다. -> 오른쪽 그림
  - [European Economic Area] <-> [zone economique europeen]



(a)

TRAIN AND TEST

TRAIN AND TEST