NEURAL MACHINE TRANSLATIONBY JOINTLY LEARNING TO ALIGN AND TRANSLATE

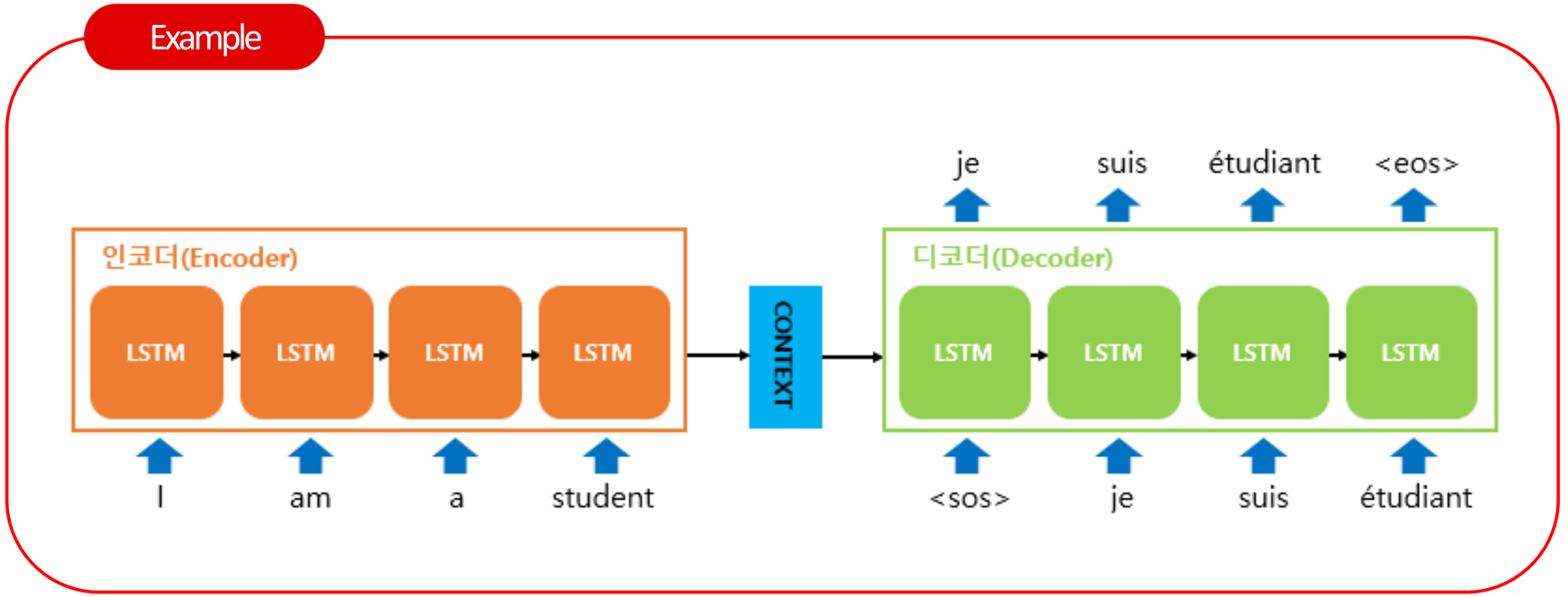
Hwang Hyeon Tae

01

Introduction

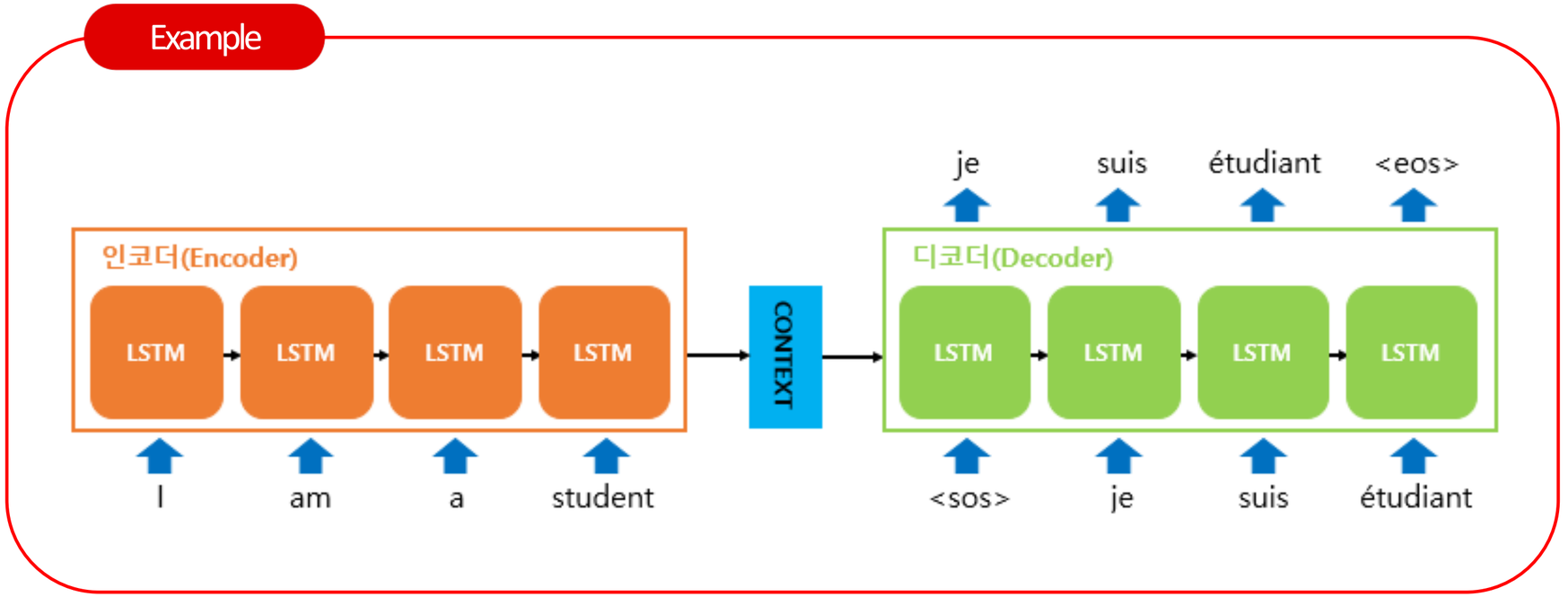
Neural machine translation

- 각 언어에 대한 인코더-디코더 구조
  - 인코더는 소스 문장을 읽고 고정 길이 벡터로 인코딩
  - 디코더는 인코딩된 벡터에서 번역 출력



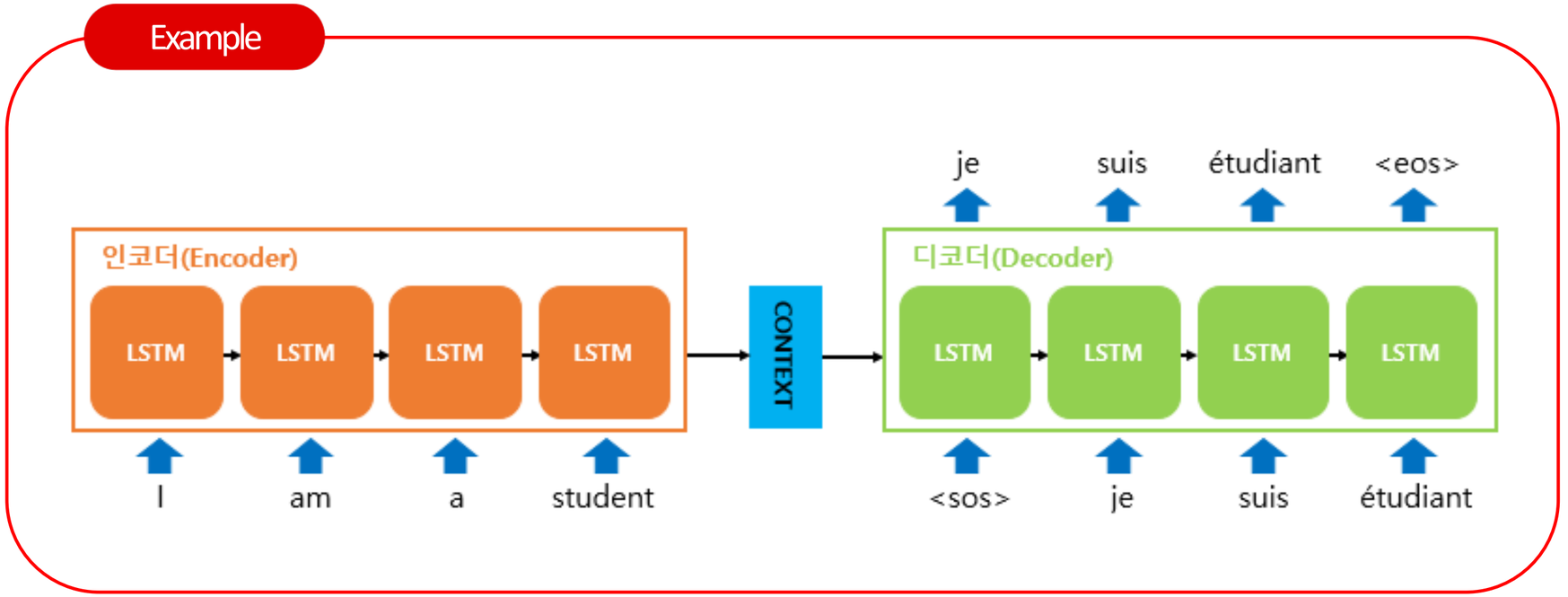
Neural machine translation

- 문제
  - 신경망이 소스 문장의 모든 필수 정보를 고정된 길이 벡터로 압축 - Bottleneck 현상 발생
    - 입력 문장의 길이가 길어질수록 인코더-디코더의 성능 저하



Neural machine translation

- 제안
  - 모델이 target 단어 예측과 관련된 소스 문장의 일부를 자동으로 검색할 수 있도록 확장
  - 입력 시퀀스를 벡터 시퀀스로 인코딩 후, 매 디코딩 step마다 벡터 시퀀스의 subset을 adaptive하게 선택
    - 문장의 모든 정보를 고정된 길이 벡터로 압축할 필요가 없다.

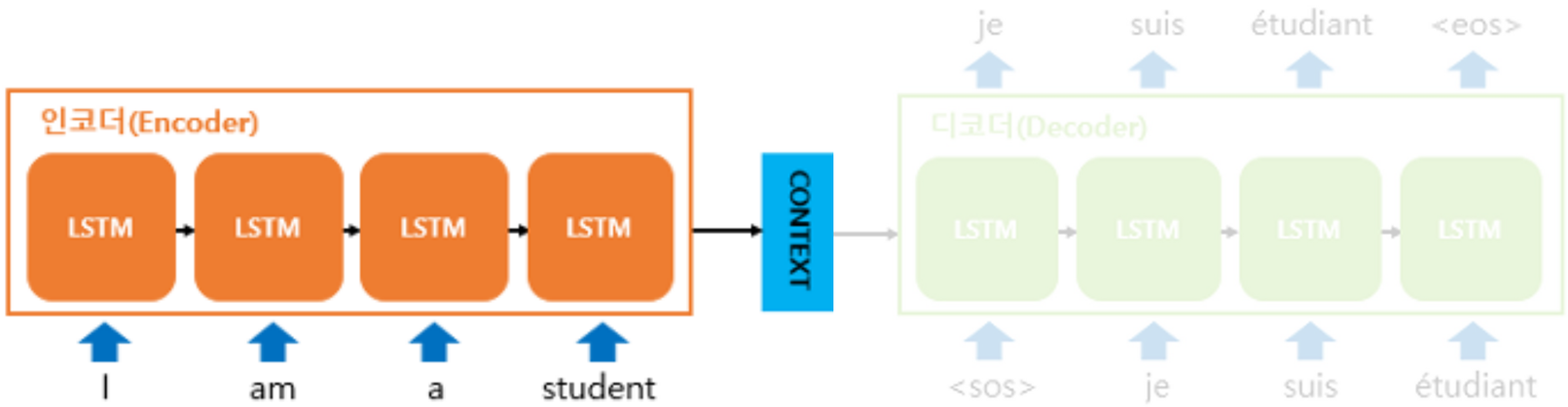


02

Background

RNN Encoder-Decoder

- Encoder



○ 일반적으로 RNN을 사용

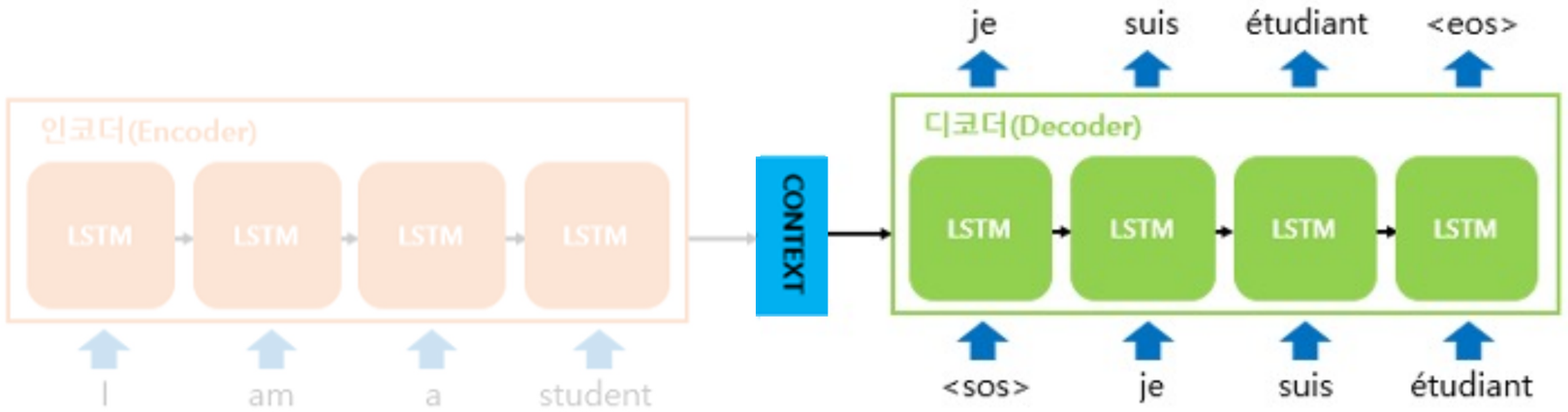
$$h_t = f(x_t, h_{t-1}) \qquad c = q(\{h_1, \cdots, h_{T_x}\})$$

Notation

h_t : hidden state
c : hidden state에서 생성된 vector (context vector)
f, q : some nonlinear functions

RNN Encoder-Decoder

- Decoder



○ c와 이전에 예측된 모든 단어 $\{y_1, \cdots, y_{t'-1}\}$ 가 주어지면 다음 단어 $y_{t'}$를 예측

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

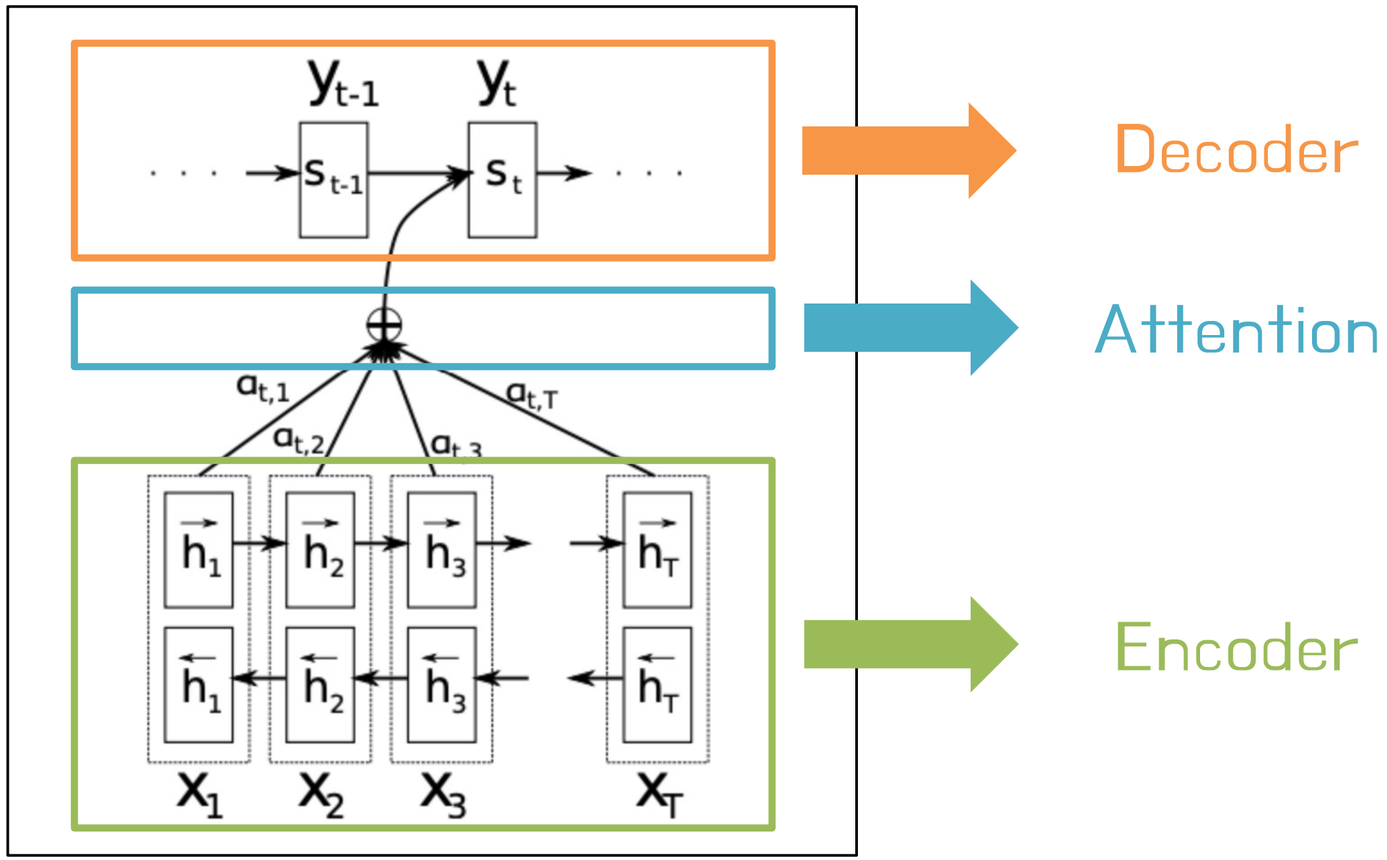
Notation

g : nonlinear, potentially multi-layered, function that outputs the probability of y_t
s_t : hidden state of RNN

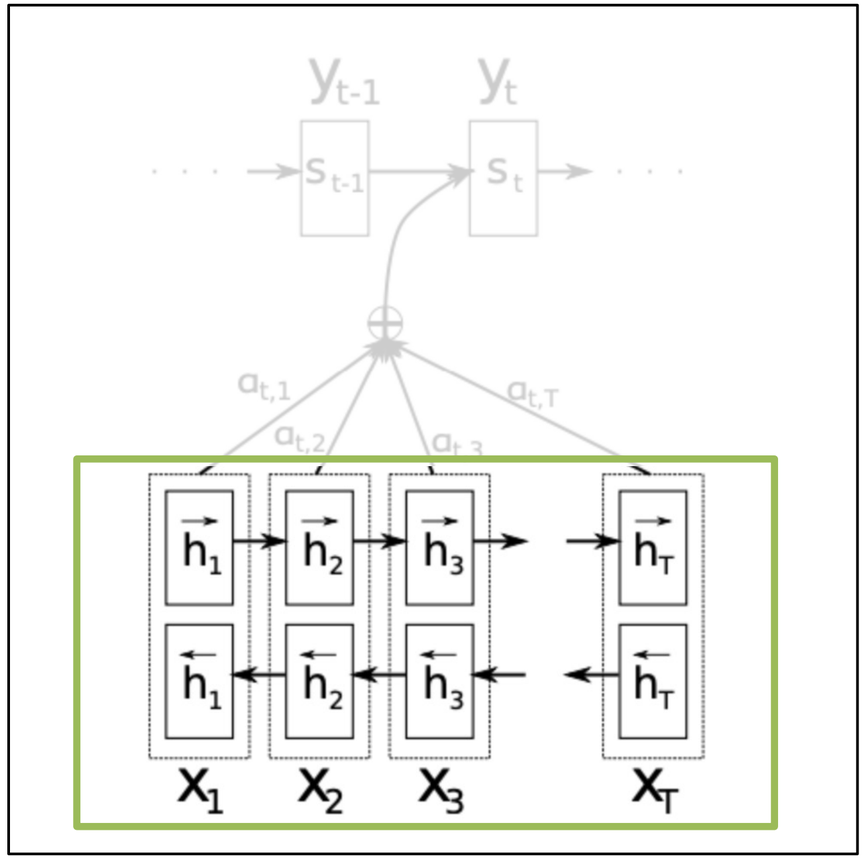
03

Architecture

Model Architecture

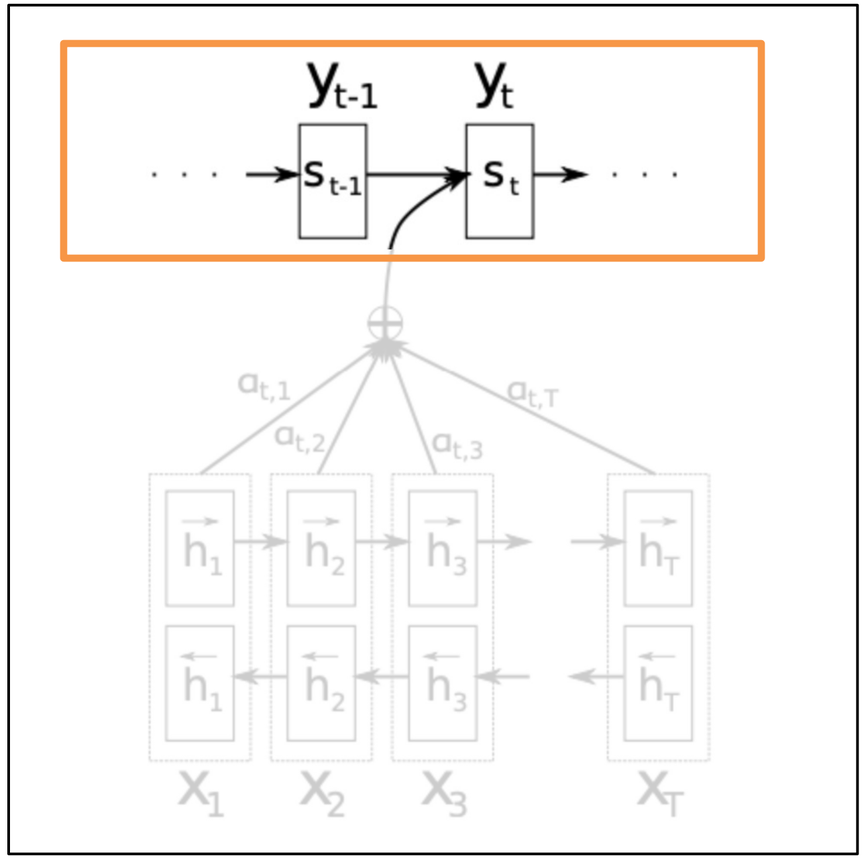


Encoder



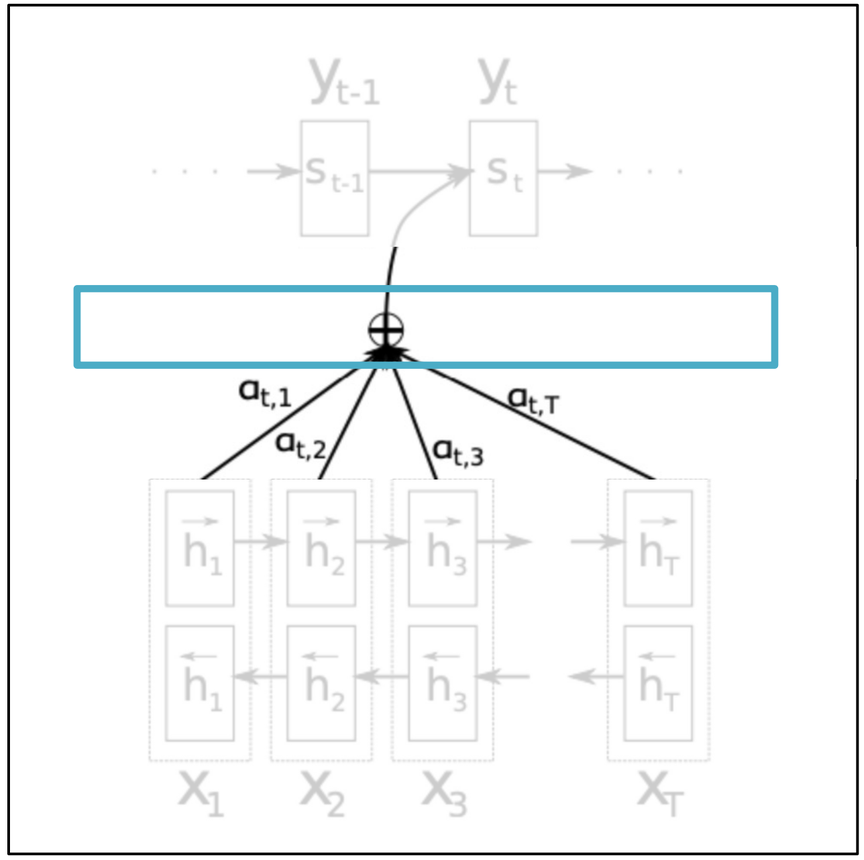
- BiRNN 사용
  - 순방향 RNN과 역방향 RNN으로 구성
  - 순방향 hidden state와 역방향 hidden state를 연결해 각 단어 x_j의 annotation h_j 생성
    - 따라서 이전 단어와 다음 단어의 summary가 포함되어, context vector를 더 잘 구성하게 된다.

Decoder



- 현 시점의 출력 y_i
  - $p(y_i|y_1,\ldots,y_{i-1},x) = g(y_{i-1},s_i,c_i)$
    - s_i : *RNN hidden state*
    - c_i : *Context Vector*
- 이 때 c_i는 Attention이 적용된 Context Vector

Attention



- Context Vector c_i
  - $c_i : \sum_{j=1}^{T_x} \alpha_{ij} h_j$
    - h_j : 입력 문장의 annotation(j번째 단어)
    - α_{ij} : weight
      - $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, e_{ij} = \alpha(s_{i-1}, h_j)$
- e_{ij}는 위치 j 주변의 입력과 위치 i의 출력이 얼마나 잘 일치하는지 점수를 매기는 정렬 모델

  -> 즉, c_i에 입력 문장 중 Attention 해야 하는 부분에 집중하여 얻은 결과를 모두 더한 값이 들어간다.

04

Experiment

Settings

- Model
  - 제안한 모델 외 비교군으로써 RNN Encoder-Decoder의 성능을 함께 보고
    - 두 모델 모두 동일한 절차와 데이터셋 사용
  - 제안된 모델 : RNNsearch
    - Encoder의 순방향 RNN과 역방향 RNN, Decoder에 각 1000개의 hidden unit 존재
  - RNN Encoder-Decoder : RNNencdec
    - Encoder와 Decoder에 각 1000개의 hidden unit 존재

- Dataset
  - WMT' 14의 News test set 사용
  - 각 언어에서 사용 빈도가 높은 단어 30000개를 목록으로 사용
    - 목록에 포함되지 않은 단어는 특수 토큰([UNK])에 매핑

Quantitative results

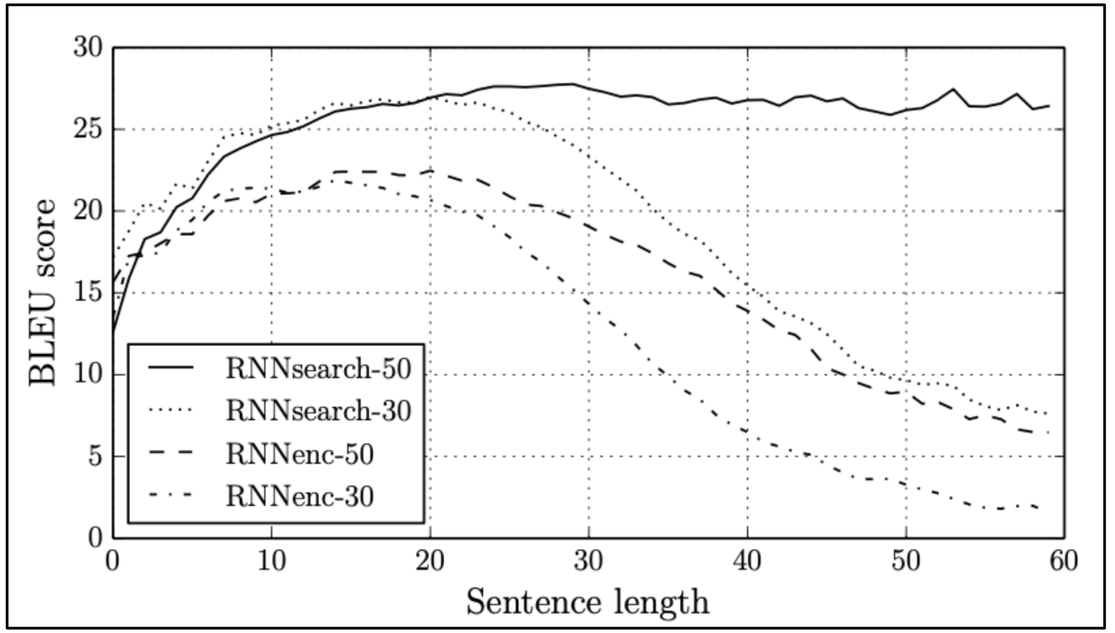
- BLEU socres of the trained models computed on the test set

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

모든 경우에 제안된 RNNsearch가 기존 RNNencdec보다 성능이 우수함을 확인

Quantitative results

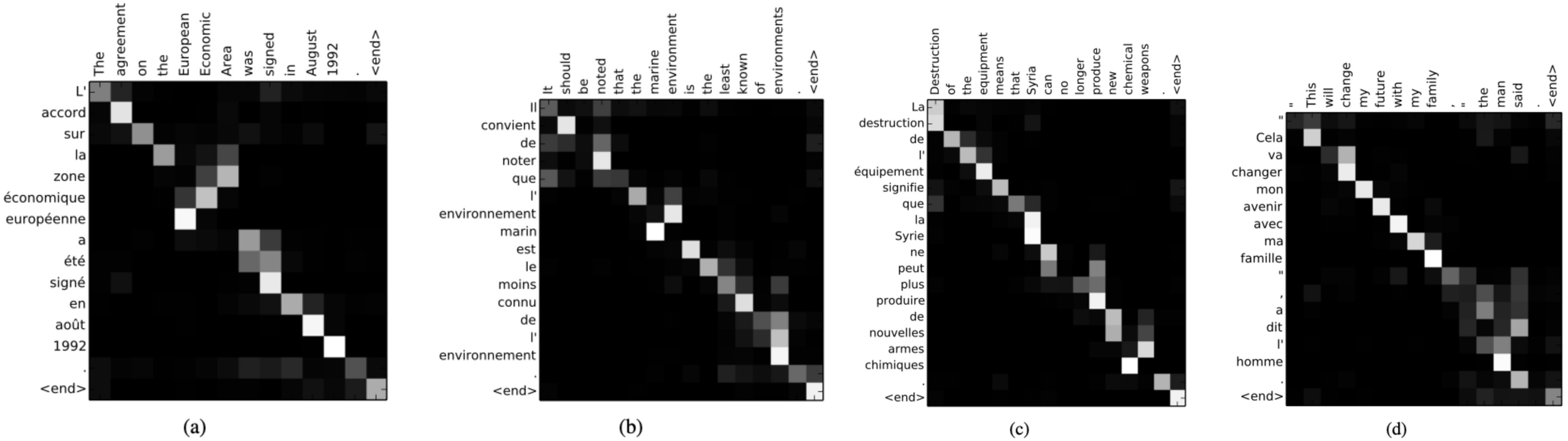
- The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences



긴 문장에서도 기존의 모델 대비 성능이 우수함을 확인

Qualitative results

- 4 sample alignments found by RNNsearch-50



Decoder에서 단어를 생성할 때마다 기존의 hidden state와 문맥을 모두 고려해 soft alignment 적용

Thank you

N. Hwang Hyeon Tae | E. gusxo3975@naver.com | L..linktr.ee/oneul_