1. Introduction

- 신경망 번역을 향상시키기 위해 attention 기법이 도입되었지만 어떤 구조가 번역에 유리한지에 대한 연구는 거의 이뤄지지 않았다. (2015)
- 이 논문에선 모든 단어를 살펴보는 global과 일부만 살펴보는 local을 살펴본다.
- state-of-the-art 달성했다.
- NMT는 도메인 지식이 적게 필요하며, 문장 길이와 무관하게 쓸 수 있어 일반화가 가능하다.
- attention 이라는 개념은 다른 양상의 데이터들 사이에도 적용시킬 수 있어서 유명해지기 시작했으며, NMT에도 성공적으로 적용되었다. 하지만 그 이상의 연구는 없었다. (2015년 기준)
- 이 논문에선 global과 local 2개의 모델을 제안한다.
  - Global
    - 전체 source words에 attend 된다.
  - Local
    - 일부 source words의 집합에만 attend 된다.
    - hard attention과 soft attention을 섞은 모델로, global model에 비해 계산량이 적고 differentiable 하다.

2. Neural Machine Translation

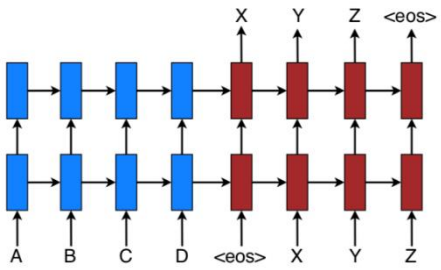

Figure 1: **Neural machine translation** – a stacking recurrent architecture for translating a source sequence A B C D into a target sequence X Y Z. Here, <eos> marks the end of a sentence.

- 보통 decoder로 RNN이 사용되지만 이 논문에선 encoder와 decoder로 stacking LSTM을 사용한다
- Encoder : source sentence를 대표하는 s를 만드는 과정
- Decoder : target 단어들을 순서대로 하나씩 만드는 과정
- s (source representation) 을 decoder의 hidden state를 initializing 하는 용도로 한번만 쓰는 경우도 있지만 이 논문에선 번역 과정 전체에 사용되며 이게 attention이다.

3. Attention-based Model

- Global과 Local로 구분한다. 모든 Source positon을 고려하는지를 본다.
- Stacked LSTM의 t번째 hidden state h_t를 input으로 받는다. 이를 통해 t번째 문맥 벡터 c_t를 얻는다. c_t는 y_t를 맞추는데 필요한 source의 정보를 저장한다.
- h_t와 c_t를 이용해서 attentional hidden state ~h_t를 얻으며, ~h_t는 소프트맥스를 통과해서 y_t의 확률을 구하게 된다.

3-1. Global attention

- c_t를 얻기 위해 인코더의 모든 hidden state를 고려한다.
- 이를 위해서 alignment vector a_t가 필요한데, a_t의 크기는 source side의 timestep과 일치한다.
- 현재의 target hidden state h_t와 source의 hidden state h_t를 비교하여 유도된다.

$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

- score는 content-based function이며 3가지 다른 방법으로 구할 수 있다.

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

- 앞서 attention based model을 만들 때는 location-based 함수를 사용했으며, alignment score는 target hidden state h_t에 가중치를 곱한 값에 softmax를 취한 값을 사용했다.
- 차이점
  - 이 논문의 모델에선 LSTM layer 맨 위층의 hidden state를 encoder와 decoder에서 사용했다.
  - Computational path가 더 간단하다.
  - a_t를 구할 때 세 가지 방법을 사용했다.

3-2. Local attention

- 전체 source word를 적용시키는 방식은 자원이 많이 필요하고 글이 길수록 실용적이지 않다.
- Local attention은 soft attention과 hard attention 이라는 개념에서 따왔다.
  - Soft attention은 global attention과 같다.
  - Hard attention은 일부분만 본다. 빠르지만 미분이 안 되는 등의 문제가 있다.
- Local attention은 문맥의 window만 고려하며 미분이 가능하다.
- 과정
  - t 시점의 target 단어에 대해 aligned position p_t를 정한다.
  - Context vector c_t는 [p_t-D, p_t+D] 사이의 source 단어들의 가중 평균이다.

- a_t의 크기는 2D+1로 고정된다.
- alignment에는 두 방식이 있다.
  - Monotonic alignment (local-m)
    - p_t = t 라고 가정한다. 같은 위치에 있는 단어끼리 연관이 클 것이라고 가정한다.
    - a_t는 global attention과 같은 방식이다.
  - Predictive alignment

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W_p} \boldsymbol{h}_t)),$$

    - W_p와 v_p는 학습되는 파라미터이며 S는 문장의 길이다.
    - Sigmoid의 결과로 인해 p_t는 0과 S 사이의 값을 가진다.
    - 해당 p_t를 기준으로 가우시안 분포를 사용한다.

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

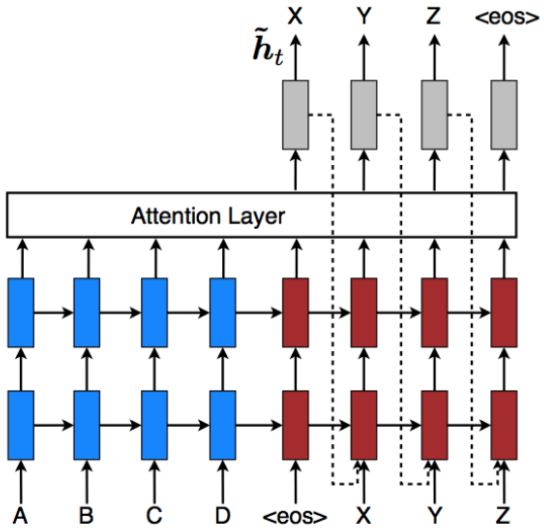

Figure 4: **Input-feeding approach** – Attentional vectors $\tilde{\boldsymbol{h}}_t$ are fed as inputs to the next time steps to inform the model about past alignment decisions.

4. Input-feeding Approach
   - Attentional decision이 독립적으로 이루어진다.
   - 본래는 어떤 단어가 번역이 되었는지 지속적으로 체크할 필요가 있다.
   - 따라서 attentional vector를 다음 시점의 input과 concat 시킨다.
     - 이전의 alignment choice를 알 수 있다.
     - 수평&수직적으로 깊은 network가 만들어진다.