Machine Translation Sequence-to-Sequence and Attention

CS224n Stanford / winter 2019 - Lecture 8 -

UOS STAT NLP Seminar

Changdae Oh

2021. 01. 02

Contents

pre-Neural Machine Translation

1950s : Early Machine Translation

Rule-based, using a bilingual dictionary to mapping

-> 두 국가의 언어간 번역 사전을 사용하여 설정된 mapping 관계에 의해 번역 수행

1990s – 2010s : Statistical Machine Translation, SMT

- 출발언어의 시퀀스가 input 되었을 때,
  Output 되는 도착언어의 시퀀스의 확률 최대화
- 두 개의 subcomponent model로 분할,
  번역모델, 언어모델 각각을 optimizing

$$\text{argmax}_y P(y|x)$$
$$= \text{argmax}_y P(x|y)P(y)$$

Statistical Machine Translation

How to learn translation model P(x | y) ?

$$P(X|Y) \rightarrow \sum_{a} P(X, a|Y)$$

a : alignment

- 풍부한 parallel data가 필요
- Alignment(번역대상 언어간 단어수준 대응관계)의 학습이 요구

Alignment 학습의 어려운 점들

- 입/출력 시퀀스 간에 대응관계인 단어가 없을 수 있음
- 일대일이 아닌 일대다, 다대일의 관계가 있을 수 있음
- 구단위의 경우 다대다 관계가 있을 수 있음
- 국가 언어간 시니피에(Signifie)의 차이 ex) 프랑스어 (빠삐용) = 한국어 (나비 + 나방)

Decoding for SMT

$$\text{argmax}_y P(x|y)P(y)$$

Language Model

Question:
How to compute this argmax?

Translation Model

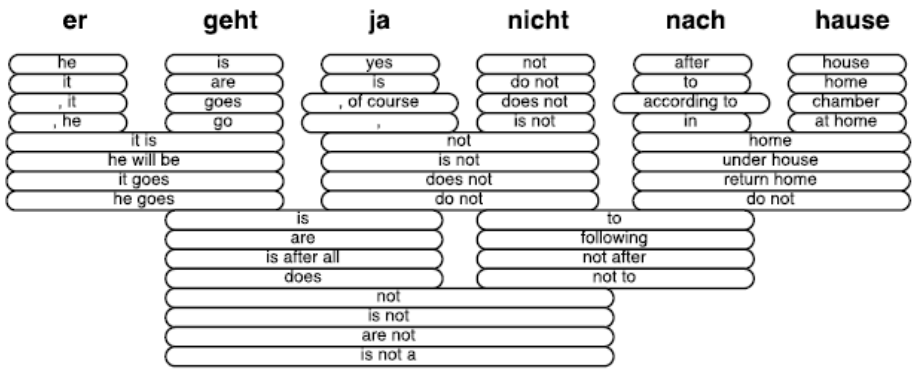
목적확률을 최대화하는 출력 y 시퀀스 결정을 위해

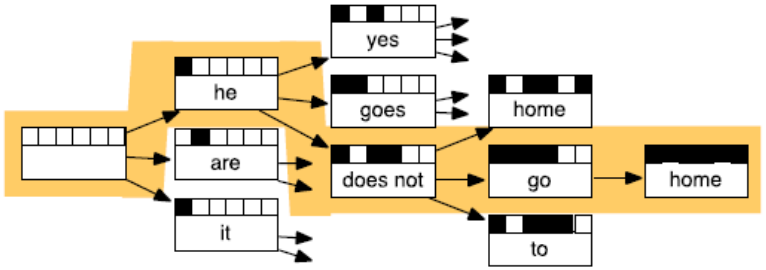
모든 가능한 시퀀스를 고려하면 좋겠지만, Too expensive

대안으로 Heuristic search algorithm을 적용하여

최선의 번역을 찾는다.

-> Decoding 과정



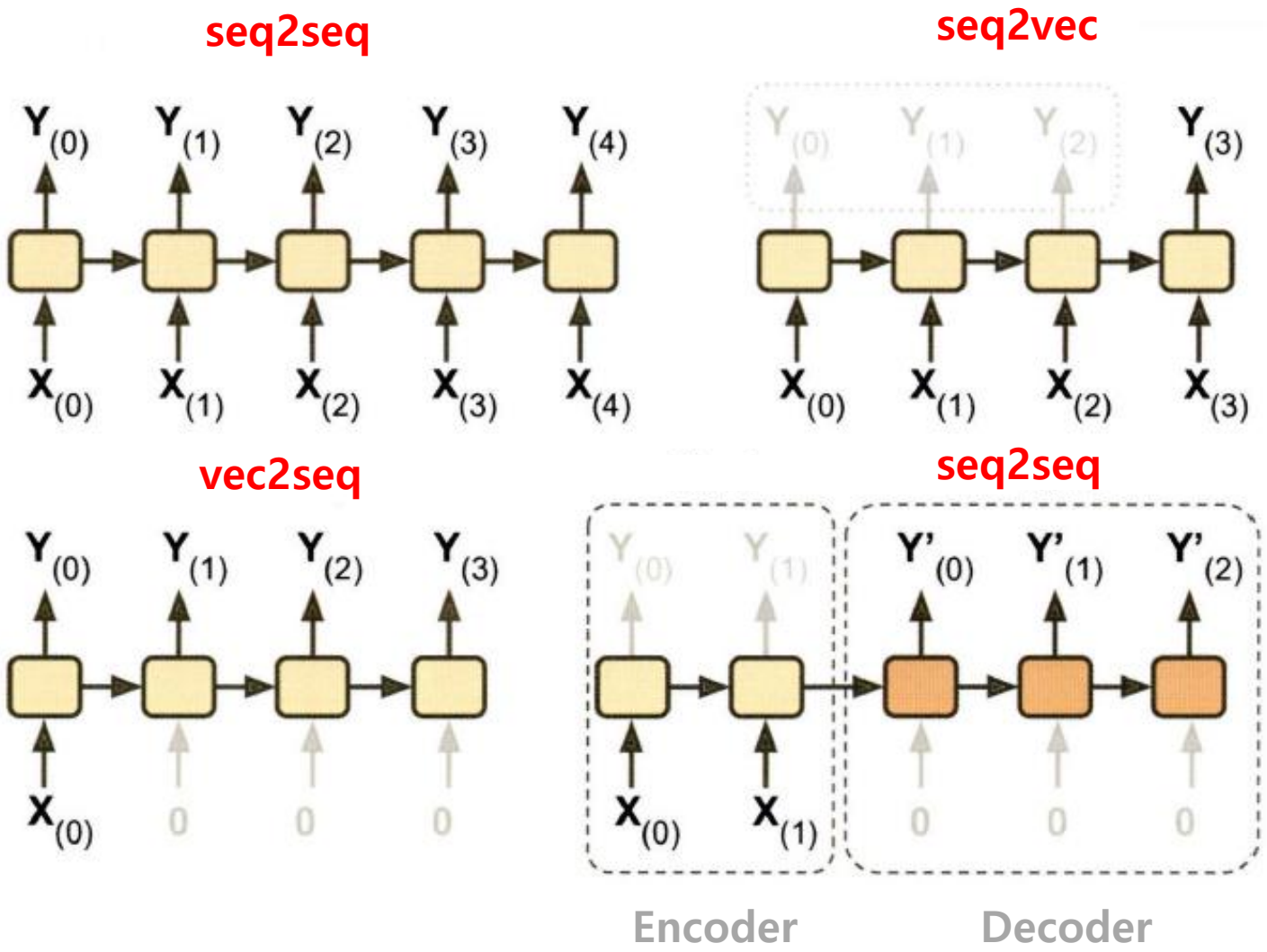


- Greedy Search
- Exhaustive Search
- Beam Search

Limitation of SMT

- 복잡하다.
- 하부요소들로 구분되어 디자인 되어있다.
- 많은 feature engineering이 요구됨.
- 인력 필요.

Types of RNN



sequence-to-sequence 모델은

seq2vec RNN과 vec2sec RNN을 연결

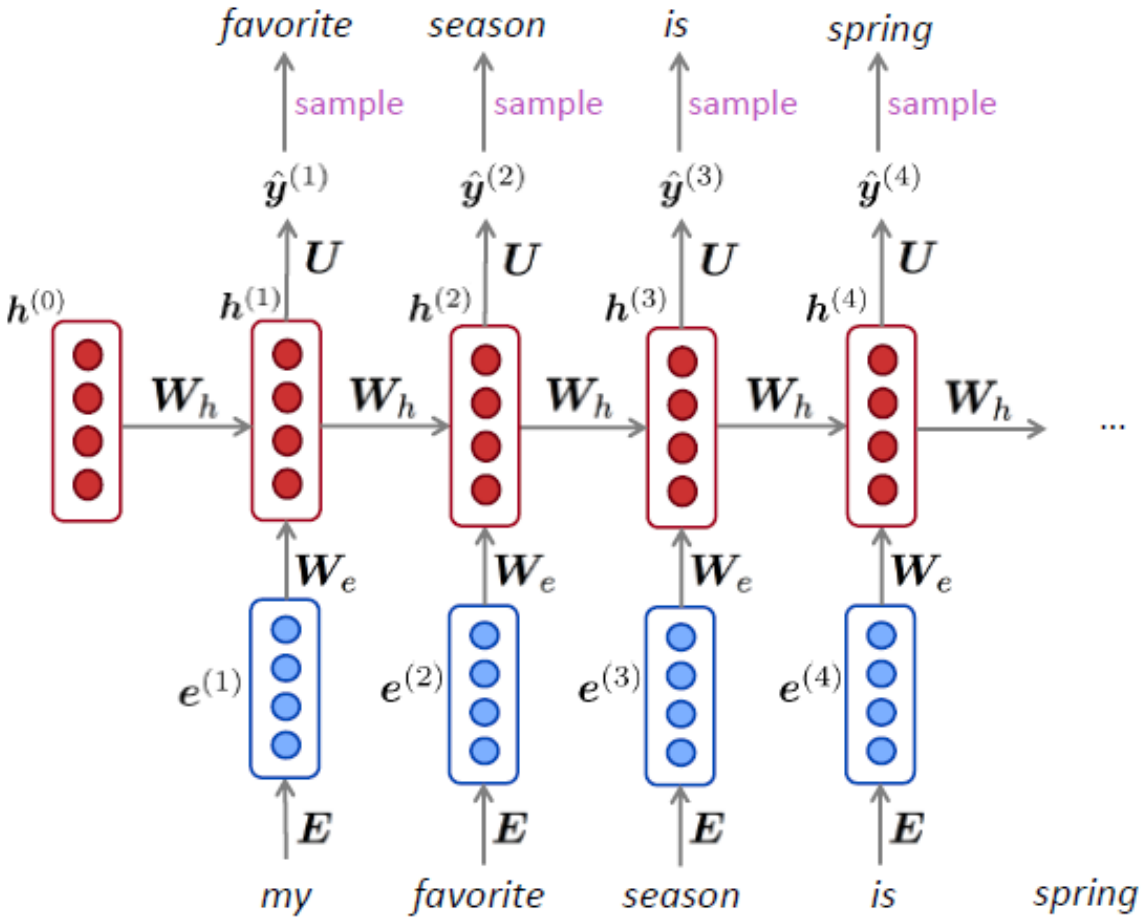
각각은 Encoder / Decoder로 불리며

- 인코더는 input시퀀스 정보 압축
- 디코더는 인코딩 된 정보를 받아

  새로운 단어 시퀀스 생성

RNN review

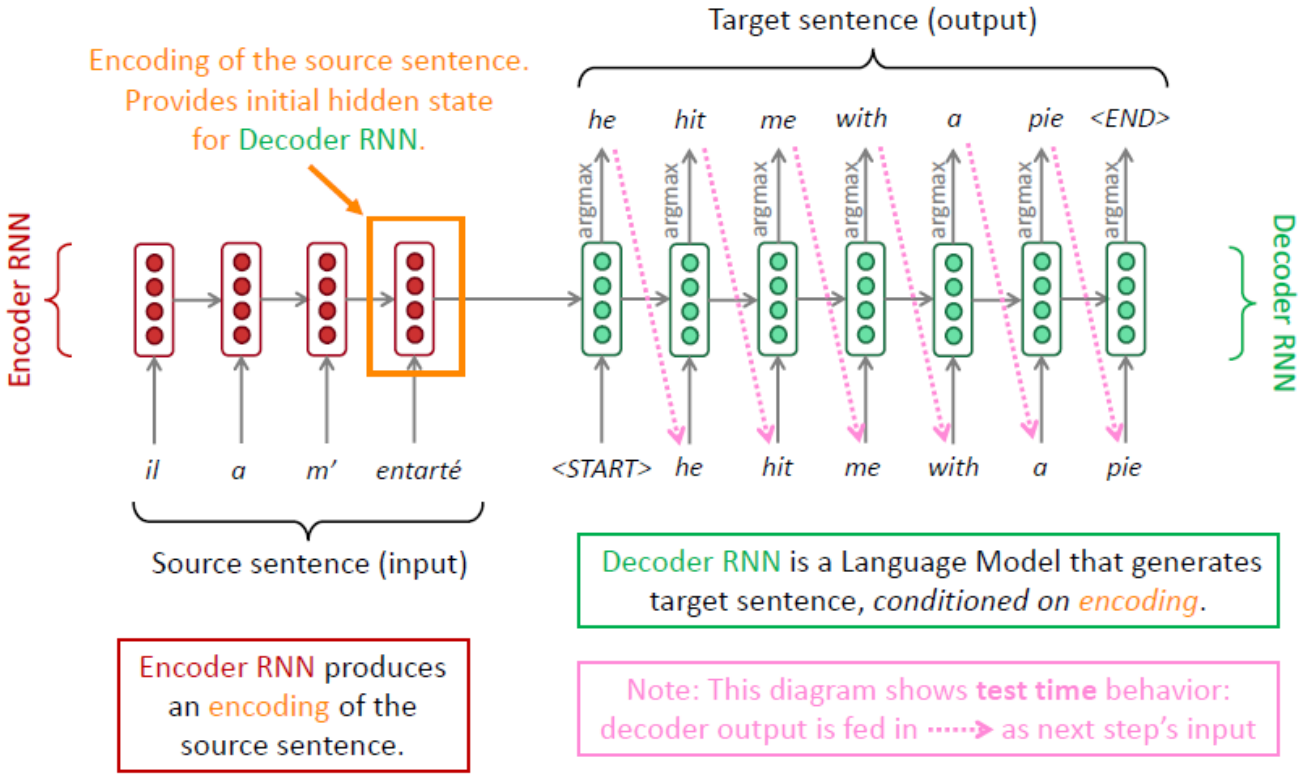
RNN을 사용한 문장 생성



1. 학습된 RNN LM에 하나의 초기단어를 input 하면 이후 출현할 단어에 대한 확률분포를 얻을 수 있다.
2. 매 스텝마다, 계산되는 다음단어의 확률분포로부터 deterministic(by argmax) 하게 혹은 probabilistic(by sampling) 하게 생성될 단어를 결정한다.
3. Training : TRUE target 시퀀스를 매 step 마다 순차적으로 input
   Testing : 매 스텝 생성단어를 다음 step의 input으로 준다.

Neural Machine Translation

sequence-to-sequence



Encoder – Decoder network

1. Parallel한 시퀀스 데이터를 seq2seq에 입력.
2. 인코더는 timestep 단위로 번역을 위한 문맥 (context) 정보를 은닉상태벡터에 누적 압축.
3. 마지막 timestep의 은닉상태벡터가 최종 문맥정보가 되어 디코더에 전달.
4. 디코더는 문맥정보를 은닉상태벡터 초기값으로 받고 <start>토큰을 timestep 시퀀스 input으로 받아 문장 생성 시작. (<end>토큰 반환시까지 반복생성)

* Training / Testing 과정에서 Decoder의 행동이 다름 (slide 7 참고)

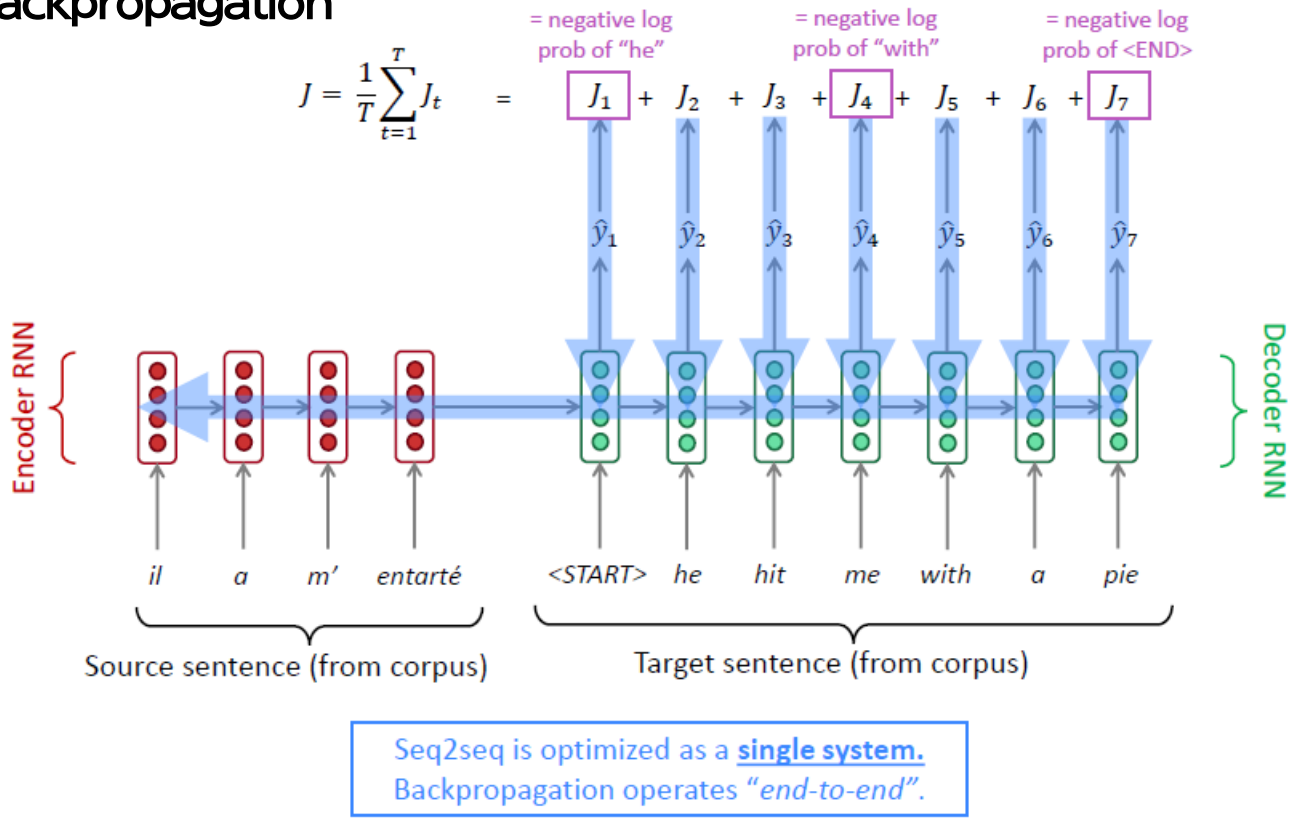
Training a NMT

NMT directly calculates $P(y|x)$:

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}$$

Probability of next target word, given target words so far and source sentence *x*

backpropagation



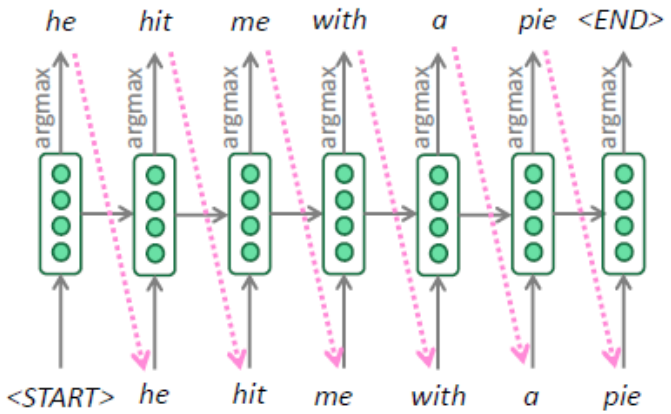
Seq2seq is optimized as a **single system.**
Backpropagation operates "*end-to-end*".

- NMT은 소스 시퀀스의 인코딩이 주어졌을 때, 다음 단어를 예측하는 일종의 조건부 언어 모델.
- SMT와는 다르게 P(y|x)를 직접 계산 가능.

- 모형에 의한 생성 시퀀스와 정답 시퀀스를 timestep단위로 비교하여 step별 Loss값을 구함.
- step별 평균 Loss를 최종 손실로 정의
- Single NN 시스템으로써 e2e로 한꺼번에 backpropagation 진행

Decoding for NMT

Greedy decoding



매 timestep마다 확률이 가장 높은 다음단어 선택

- 최적해를 보장하지 않음.
- Cost 낮음.

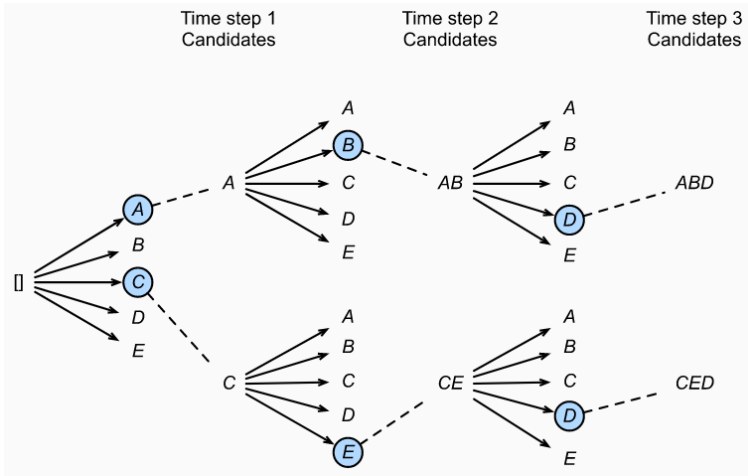
Exhaustive search

그냥 다해보기

가능한 조합 모두 생성 후 확률비교, 확률이 가장 높은 문장 선택

- 최적해를 보장 함.
- Cost 매우 큼.

Beam search



각 step마다 확률기반 측도로 Score 계산. Beam size k만큼의 후보 유지, 나머지 제거.

- 최적해를 보장하지 않음.
- 배제성을 줄여 안정적.
- Cost 보통.

NMT의 장단점 및 평가

장점

- 높은 성능
- 단일 NN 모델로써 E2E 최적화가 진행됨
  (SMT는 translation model,
  language model 따로따로 최적화)
- 인적 노력 덜필요

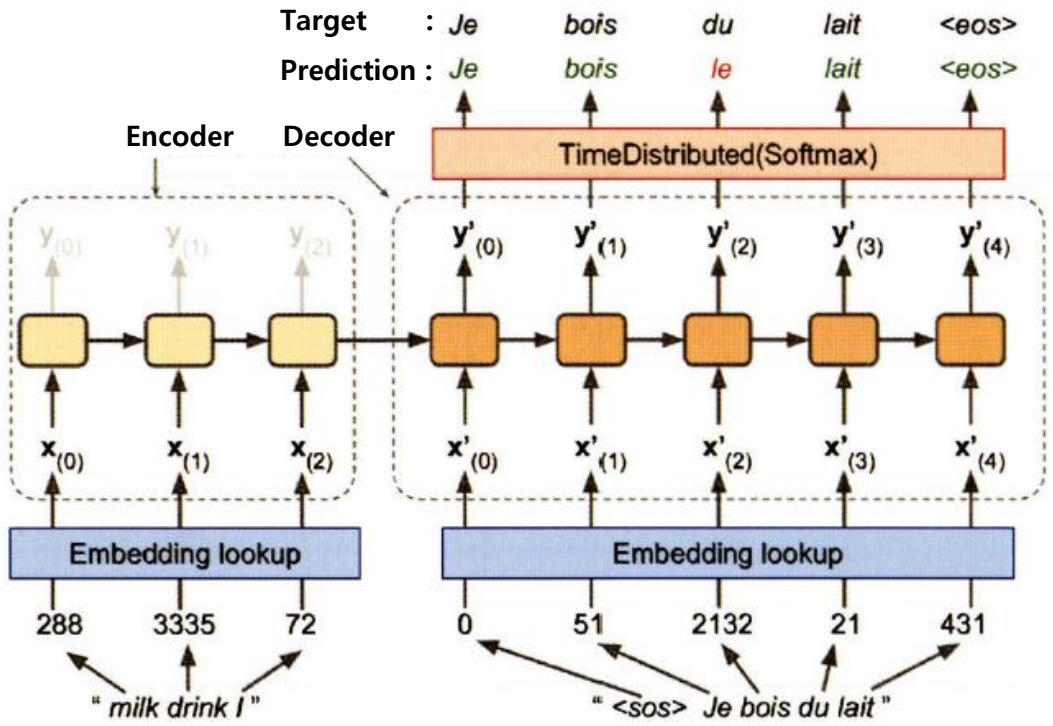
단점

- 신경망 모델이다보니 해석 어려움
- 세부적인 제어, 튜닝이 힘듬

BLEU 기준 큰 성능향상이 있었으나
아래와 같은 **한계점** 존재

- OUT of vocab 문제
- Train/test Domain 불일치 문제
- 긴 text에 대한 문맥유지 문제
- 풍부한 쌍 시퀀스 얻기 힘듬

Problem of vanilla seq2seq



다루는 시퀀스의 길이가 길어질수록

아래와 같은 문제점들이 부각된다.

- Information bottleneck

  (고정된 길이의 은닉상태벡터에

  모든 문맥정보를 보존하기는 힘듬)

- **장기 의존성 식별의 어려움**

  (고질적인 RNN의 문제)

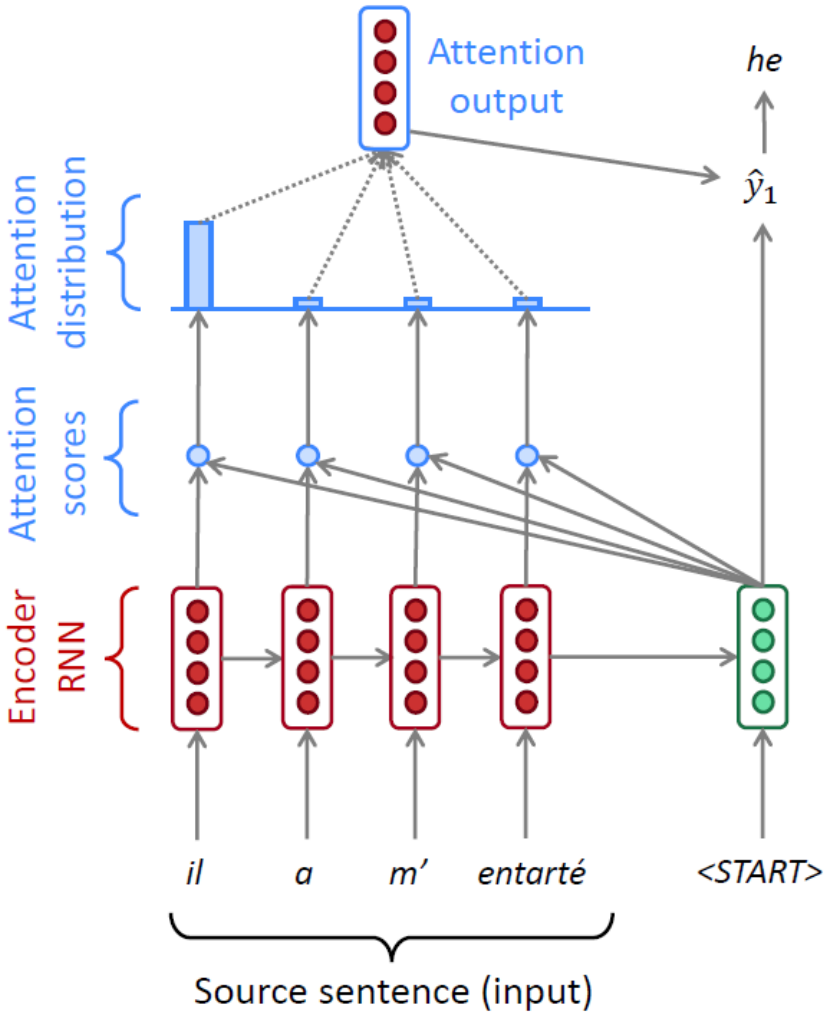
- Backpropagation 과정에서

  거쳐가야 할 경로가 길어진다.

  (gradient vanishing)

Attention mechanism

Improvement of seq2seq



Core idea :

- **출력 시퀀스의 단어와 유사도가 높은 입력 시퀀스의 단어**에 대한 정보를 추려내자.
- Decoder가 Encoder의 마지막 은닉상태만 이용하지 않고, 매 timestep마다 Encoder의 모든 은닉상태를 참고하자.

Details :

매 step마다 아래와 같은 과정으로 Attention score / output 계산

- Attention scores : 모든 시점 encoder의 은닉상태벡터들과 decoder의 현재시점 은닉상태벡터를 각각 내적(유사성 측도).
- Attention weight : 어텐션 점수를 softmax 계층에 통과시키고 0과 1사이 확률값으로 정규화하면, 어텐션 가중치벡터를 얻음.
- Attention output : 어텐션 가중치벡터와 encoder의 은닉상태벡터들의 weighted sum을 통해 input 시퀀스의 맥락(context)요약벡터 Attention output을 얻음.

Attention mechanism

< equations >

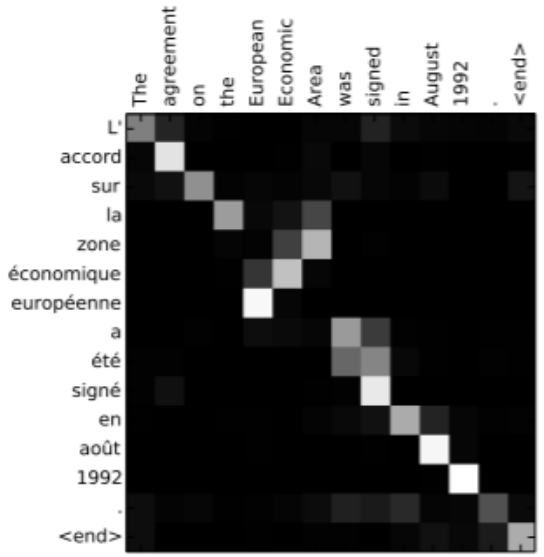
$h_1, \ldots, h_N \in \mathbb{R}^h$ $s_t \in \mathbb{R}^h$		– Encoder의 은닉상태벡터들 – timestep t에서 Decoder의 은닉상태벡터
$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$	step t Attention 점수벡터	Encoder의 은닉상태벡터들과 Decoder의 step t 은닉상태벡터를 내적
$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$	step t Attention 가중치벡터	Step t Attention 점수벡터에 softmax 함수를 취함.
$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$	step t Attention 결과벡터	step t Attention 가중치벡터와 Encoder의 은닉상태벡터들 가중합
$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$		Step t Attention 결과벡터와 step t Decoder의 은닉상태벡터 병합.

Benefits of Attention

- NMT performance 향상
- Bottleneck 문제 해소

  (모든 시점, 디코더가 direct하게 source 시퀀스와 접촉)
- Vanishing gradient 문제 해소

  (긴 step을 거쳐 역전파되는 대신 Attention 연산에 의한 shortcut이 생성되었음.)
- 해석력 증가



Attention weight의 해석

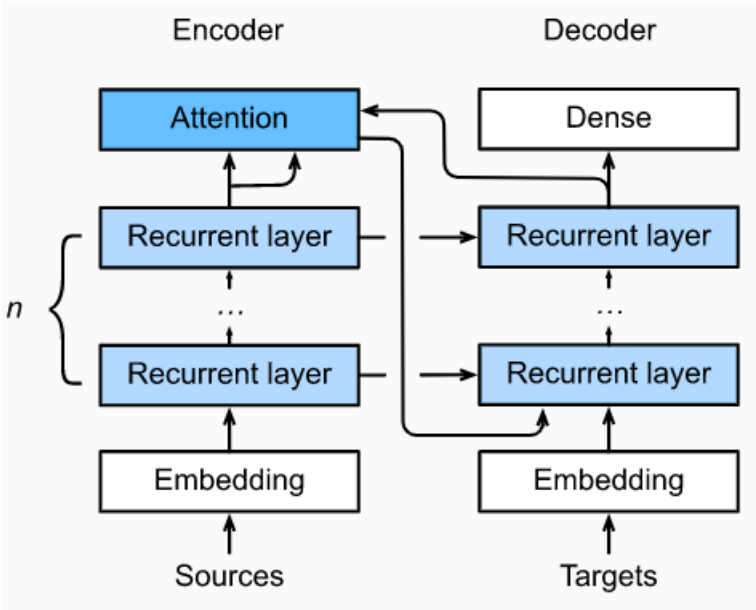
벡터 내적(dot product)을 거쳐 계산된
Attention score와 weight에는
Encoder와 Decoder내부 단어들 간에 존재하는
Alignment에 대한 정보가 포함되어 있다.
(확률에 의해 표현되는 soft-Alignment)
-> 어텐션을 통해 network가 대응관계를 스스로 학습

Improvements & variants

- 강의에서는 일방향 seq2seq로 설명하였으나 양방향으로도 확장 가능하다.
- Vanilla RNN 층 대신 그것의 변형인 LSTM, GRU 등도 사용할 수 있다.
- 표현력을 더 높이기 위해 RNN 계층을 더 깊게 쌓을 수 있다.

  -> 이 경우 구현단계에서 Attention 연산을 수행하는 계층을 이동시켜

  Attention output을 다른 RNN 계층이 참조하도록 변형할 수 있다.



어텐션 연산 수행 시

내적외에 다른 연산들을 고려해볼 수 있다.

Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$

Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$