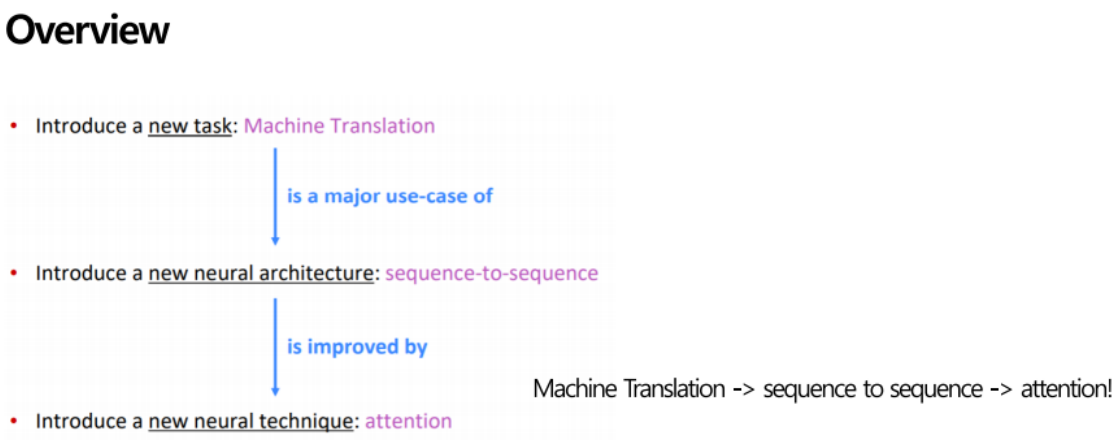
Lecture 7 & 8

작성일시	@2022년 2월 4일 오전 3:46
최종 편집일시	@2022년 2월 4일 오후 5:06
문서 유형	
상태	
작성자	
최종 편집자	
이해관계자	

Contents



Machine Translation

Machine Translation

x: *L'homme est né libre, et partout il est dans les fers*

↓

y: *Man is born free, but everywhere he is in chains*

MT는 한 (소스) 언어의 문장을 (목표) 언어의 문장으로 번역하는 것.

- Input sentence → Output sentence

1950's cold war (Early machine translation)

1950`s : Early Machine Translation

Machine Translation research began in the early 1950s.

- Russian → English (motivated by the Cold War!)

- "Rule-based", 두 개 언어의 사전을 통해 각 단어가 대응되는 것을 찾아서 번역
- Russian → English

1990's - 2010's (Statistical Machine Translation)

1990s-2010s: Statistical Machine Translation

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

Translation Model

Models how words and phrases should be translated (*fidelity*). Learnt from parallel data.

Language Model

Models how to write good English (*fluency*). Learnt from monolingual data.

1. Translation Model : 작은 단어와 구의 번역
2. Language Model : 좋은 문장, 좋은 구조 도출

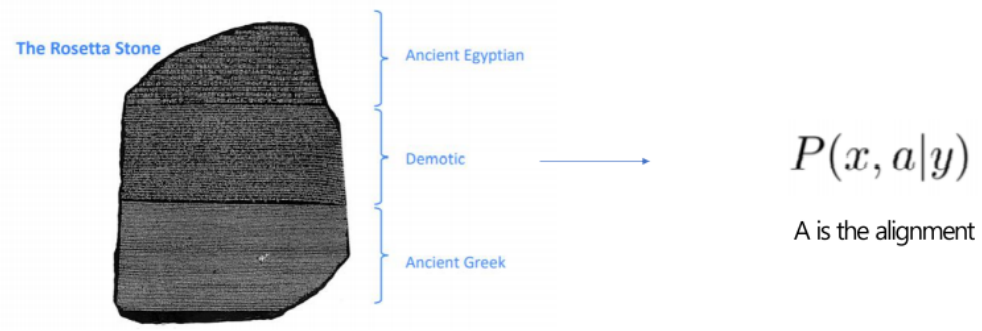
- 기존의 rule-based 방식이 아닌 data를 통해서 번역

  → 데이터를 통해 확률 분포를 학습

- 특정 언어 input sentence x에 대해서 또 다른 언어 output sentence인 y로 번역할 때 다음의 확률이 가장 높은 sentence를 선택
- Bayes theorem을 사용

Translation Model → Parallel Data / Language Model → Monolingual Data

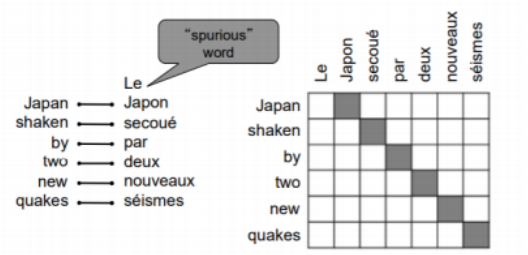
Learning alignment for SMT



- 많은 양의 병렬 데이터가 필요 → Alignment !

Alignment

What is alignment?

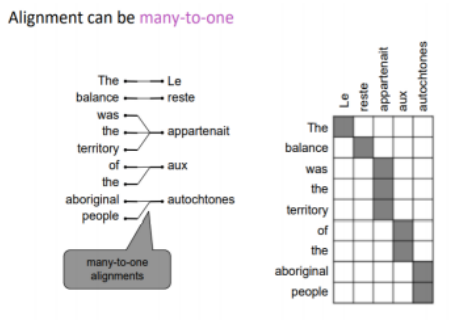


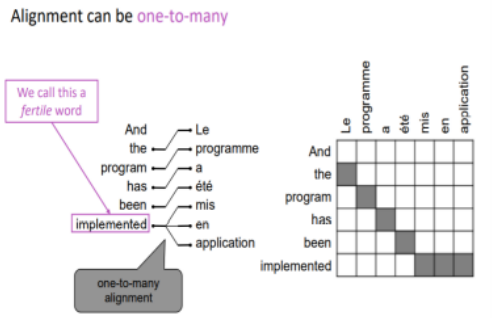
1. 정렬이란, 두 문장 사이에서 특정 단어쌍들의 대응
2. 어떤 단어들은 대응되지 않을 수도..

- 병렬 처리된 두 문장에서 특정 단어쌍들의 대응 관계

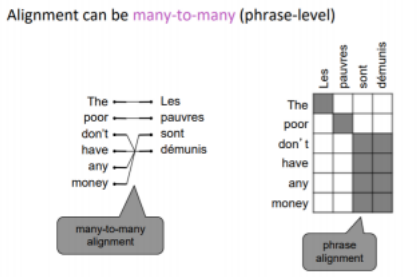
- 문제점 - 각 언어들의 특성이 다름 ex) 단어들의 순서, 품사의 순서 → 일대일 대응이 어려움

Alignment is complex





Alignment is complex, how learn?

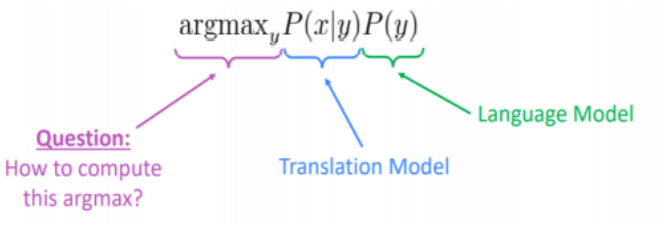


How learning?

How to learn

- 무차별 대입 솔루션

Decoding for SMT



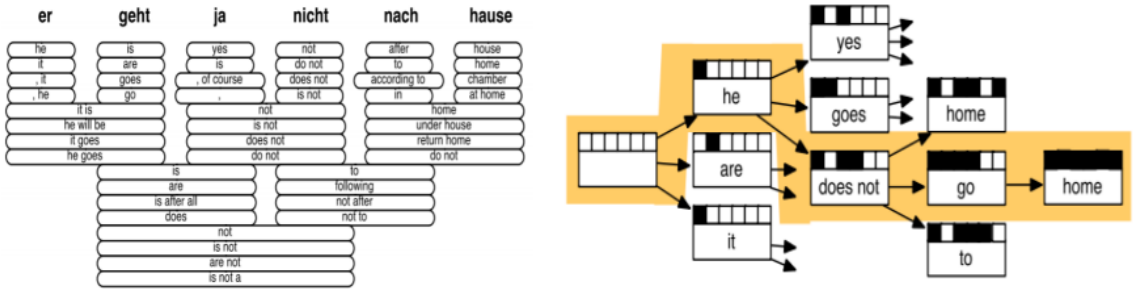
1. 무차별 대입 솔루션
2. Heuristic 알고리즘

→ 가능한 모든 y를 열거하여 모든 확률을 계산

: 시간과 비용 모두 비효율적

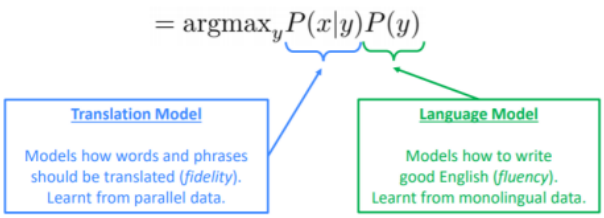
- Heuristic Algorithm

Decoding for SMT



→ 너무 낮은 확률들은 계산하지 않고 높은 확률 위주의 가지들로 선택

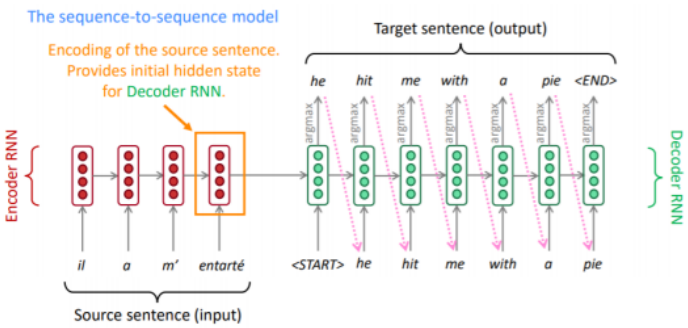
1990s-2010s: Statistical Machine Translation



1. 좋은 성능을 내지만 매우 복잡한 구조
2. 각 system은 각 부분으로 나눠서 sub-system들이 모여 있는 형태
3. 많은 feature engineering이 필요
4. 추가적인 많은 자료 필요
5. 사람의 손을 많이 거쳐야함

Neural Machine Translation (Sequence to Sequence)

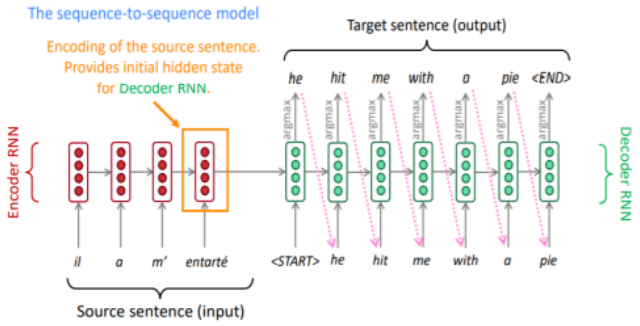
Neural Machine Translation (NMT)



1. 단일 신경망으로 기계번역을 하는 방법
2. 두개의 RNN을 포함한 seq-to-seq
3. 언어모델인 Decoder RNN은 target sentence를 생성
4. Output이 다음 단계의 input이 됨.

- 두 개의 RNN(Encoder RNN, Decoder RNN(조건부 언어 모델)을 포함)
- 소스 문장은 Encoder RNN에 투입 → 인코딩 후 Decoder RNN으로 이동 → 인코딩에 따라 출력값 획득, 다음 나올 단어의 확률분포 argmax 취하기 → Decoder에 전 단계 단어 투입 후 다시 피드백, argmax 취한 후 단어 맞추기 → 종료 토큰 생성 시까지 반복

Sequence-to-sequence is versatile!



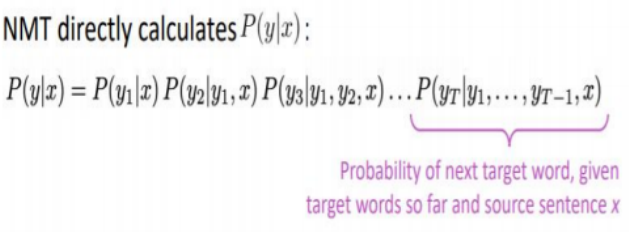
1. Summariztion (long text -> short text)
2. Dialogue (previous utterances -> next utterance)
3. Parsing (input text -> output parse as sequence)
4. Code generation (natural language -> python code)

- 긴 텍스트를 짧은 텍스트로 요약 가능

- 전의 단어가 다음 단어로 이어져 문맥 파악 가능
- input text가 일련의 output parse로 parsing
- 자연어 코드 생성 가능

How to learn

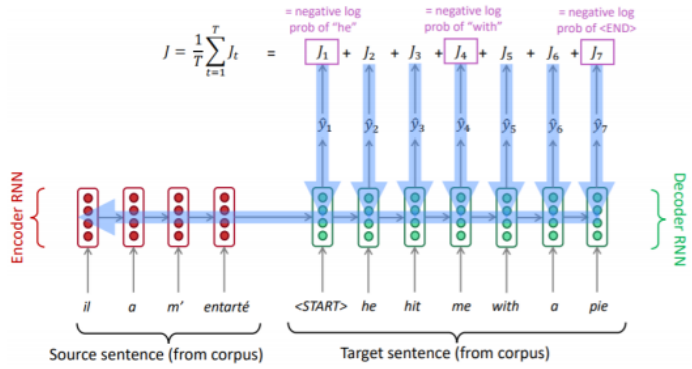
Neural Machine Translation (NMT)



요약하자면 조건부 언어모델!

- “조건부 언어 모델”
- “어떻게 NMT 모델을 학습시키는가?” → 병렬 말뭉치 !

Training a Neural Machine Translation system



End- to-End
시스템 전반적으로 최적화!

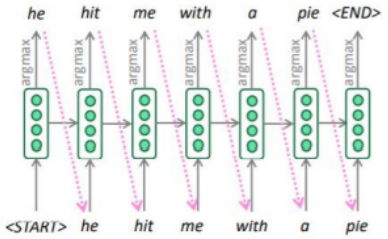
- 시스템을 전반적으로 최적화하기, ‘END-TO-END’ (문장 요소를 모두 고려)
- 인코더 RNN에 문장 삽입 → 디코더 RNN의 모든 단계에서 다음에 올 확률분포 Y HAT의 손실(Loss entropy, etc)을 계산 → 모두 구하고 더해 단계 수로 나누어 최종 LOSS

값 획득

- 역전파는 전체 시스템에 걸쳐 흐름
- 미리 정의된 최대길이에서 최대한 짧은 문장을 채우는 방법으로 진행

Problem

Greedy decoding



- Greedy decoding has no way to undo decisions!
  - Input: *il a m'entarté* *(he hit me with a pie)*
  - → *he* ____
  - → *he hit* ____
  - → *he hit a* ____ (whoops! no going back now...)

- 각 단계에서 가장 확률이 높은 단어를 도출 → 결정 수정 불가

Solution

1. Exhaustive search decoding

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

2. Beam search decoding

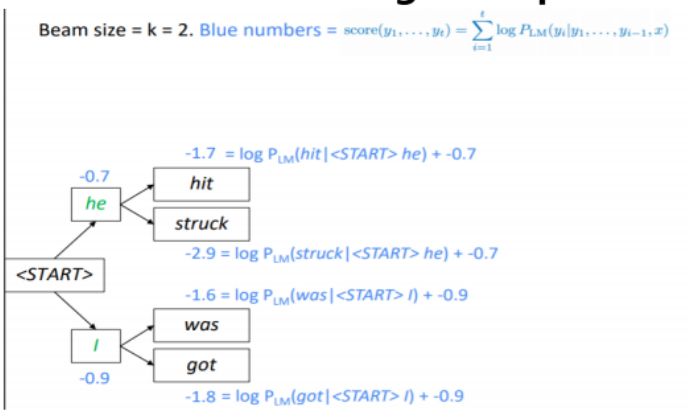
$$\text{score}(y_1,\ldots,y_t) = \log P_{\text{LM}}(y_1,\ldots,y_t|x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1,\ldots,y_{i-1},x)$$

1. 철저하게 가능한 모든 언어 번역 공간 검색 → 디코더의 각 단계 t에서 가능한 부분 번역의 거듭제곱으로 v(어휘의 크기)를 취적 = 모든 경우의 수 계산
2. 빔 서치 - 디코더의 각 단계에서 k 개의 가장 가능성이 높은 부분 번역 선택 후 가지치기

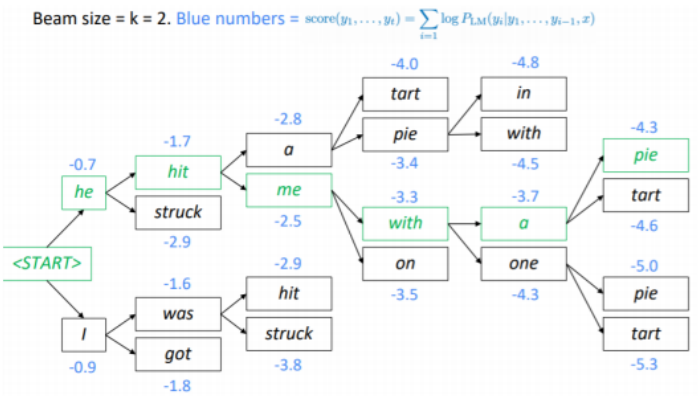
= 가설의 점수가 가장 높은 번역 선택 → 최적의 솔루션 불문명 but 효율적

#Example

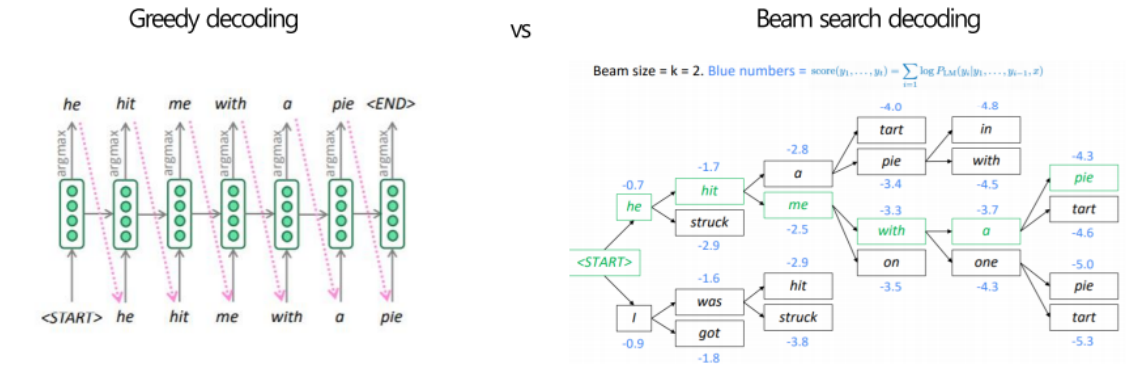
Beam search decoding: example



Beam search decoding: example



Beam search decoding: stopping criterion



Greedy : 모델이 end token을 생성할때까지 decode를 진행

Beam Search : 한 가설을 end token을 생성하고 나면 그대로 두고, 다른 가설들을 탐색

→ 한계를 설정해줘야 함

- 미리 정한 시간 단계 t에 도달하면 중지 / n개의 완료된 가설을 만들면 중지
- 가설 모음 중 가장 높은 점수를 받은 것을 선택 → 긴 가설이면 낮은 점수를 받게 된다는 문제

Beam search decoding: finishing up

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

$$\frac{1}{t} \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- t 수로 나누어 평균을 구하고, 그 중 최상위 선택

Advantages, Disadvantages of NMT

Advantages

1. 더 나은 성능
2. Single neural network to be optimized end-to-end (하부구조가 개별적으로 optimized될 필요 X)
3. 인간의 노력 덜 필요

Disadvantages

1. Hard to debug
2. Difficult to control

- Feature Engineering 불필요, 모든 언어쌍에 같은 방법으로 적용 가능
- 하부구조 존재 x → 특정 오류 발견 후 수정/제어할 사후규칙 수립에 어려움

BLEU

How do we evaluate Machine Translation? BLEU (Bilingual Evaluation Understudy)

- 예측된 sentence : 빛이 쐬는 노인은 완벽한 어두운곳에서 잠든 사람과 비교할 때 강박증이 심해질 기회가 훨씬 높았다
- true sentence : 빛이 쐬는 사람은 완벽한 어둠에서 잠든 사람과 비교할 때 우울증이 심해질 가능성이 훨씬 높았다

- 1-gram precision: $\frac{\text{일치하는 1-gram의 수(예측된 sentence중에서)}}{\text{모든 1-gram'항 (예측된 sentence중에서)}} = \frac{10}{14}$
- 2-gram precision: $\frac{\text{일치하는 2-gram의 수(예측된 sentence중에서)}}{\text{모든 2-gram'항 (예측된 sentence중에서)}} = \frac{5}{13}$
- 3-gram precision: $\frac{\text{일치하는 3-gram의 수(예측된 sentence중에서)}}{\text{모든 3-gram'항 (예측된 sentence중에서)}} = \frac{2}{12}$
- 4-gram precision: $\frac{\text{일치하는 4-gram의 수(예측된 sentence중에서)}}{\text{모든 4-gram'항 (예측된 sentence중에서)}} = \frac{1}{11}$

$$(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}} = (\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11})^{\frac{1}{4}}$$

$$BLEU = min(1, \frac{output\ length(\text{예측 문장})}{reference\ length(\text{실제 문장})})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

$$= min(1, \frac{14}{14}) \times (\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11})^{\frac{1}{4}}$$

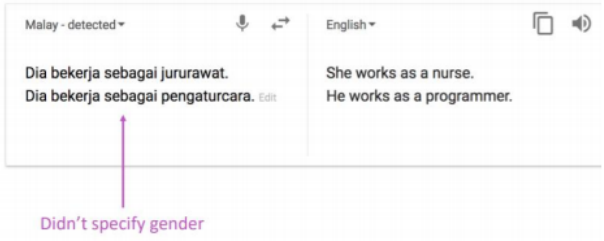
- 유사도 측정: 기계에 의해 번역된 문장과 사람이 작성한 문장 비교

의의와 한계

- Google : 소수의 엔지니어로 몇 달만에 더 나은 성과

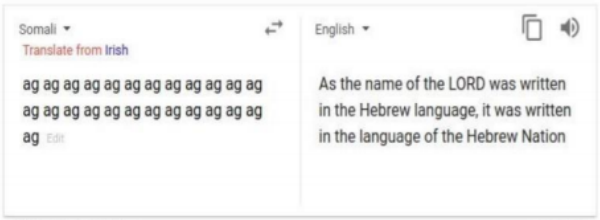
- 단어의 범위를 초과하면 목표 단어 생성x (학습되지 않은 단어)
- 학습데이터와 test 데이터 사이의 domain 불일치 → 성능 저하
- 한 문장 encoding을 하는 것이기에 긴 text의 경우 context 유지에 어려움
- 리소스가 부족한 언어 쌍의 경우 성능 저하 문제
- 관습적인 언어(상식) 이해도 떨어짐
- 편향된 번역 존재

So is Machine Translation solved?



- 반복된 문장 멋대로 해석

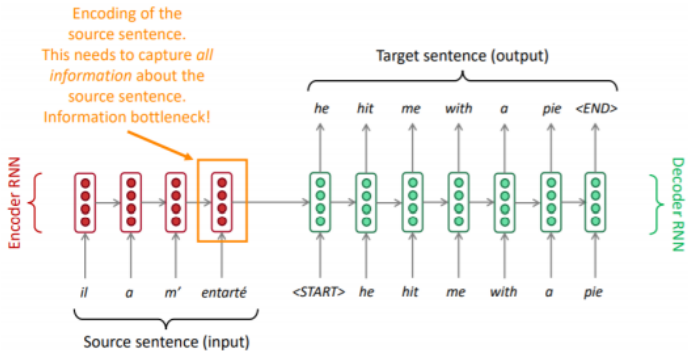
So is Machine Translation solved?



→ ATTENTION

Attention

Sequence-to-sequence : the bottleneck problem

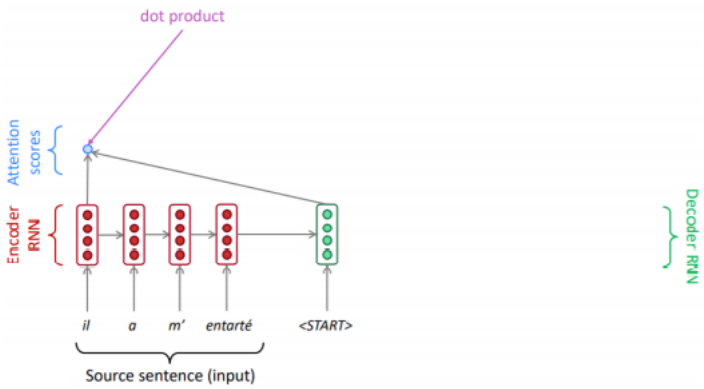


맨 끝에서 모든 정보를 캡쳐 강요
→ 너무 많은 압력 → 병목문제

- seq-2-seq : 문장의 정보가 끝에서 다 인코딩 되어버리는 병목현상 발생
- Attention → 단계별 집중!

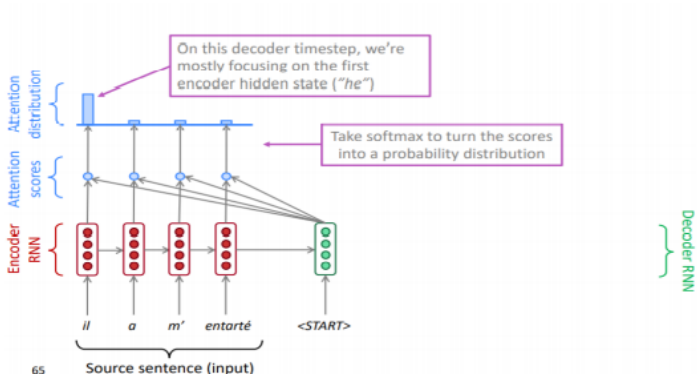
Phase

Sequence-to-sequence with attention



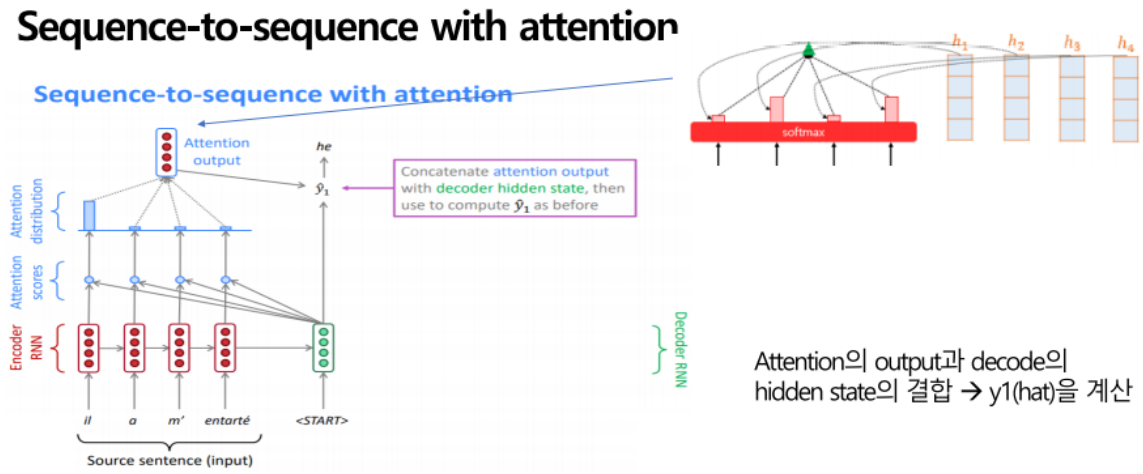
- 인코더의 첫 단계와 내적, 스칼라 값 추출 → attention score 생성
- 현재 디코더의 시점 t에서 단어를 예측하기 위해 인코더의 모든 은닉 상태 각각이 디코더의 현 시점의 은닉상태 states와 얼마나 유사한지를 판단

Sequence-to-sequence with attention



모든 인코더의 step마다 반복!
Attention을 주는 것

- attention score를 소프트맥스에 넣고 확률 분포를 얻음 → 인코더 hidden states의 가중치



- Attention 분포와 인코더의 hidden states를 가중 합하여 attention output을 뽑고, 그리고 그 attention output과 디코더의 hidden state를 결합하여 y1(hat)이라는 결과 he를 도출

Attention : in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores e^t for this step:

$$e^t = [s_t^T h_1, \ldots, s_t^T h_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(e^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output a_t

$$a_t = \sum_{i=1}^{N} \alpha_i^t h_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output a_t with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

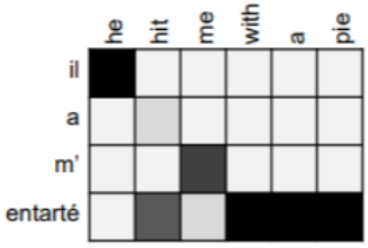
$$[a_t; s_t] \in \mathbb{R}^{2h}$$

73

1. Encoder hidden states
2. Decoder hidden state
3. Softmax
4. Attention output
5. Y hat

Effect

Attention is great



1. NMT 성능을 향상시킴
2. 병목문제 해결
3. 기울기 소실 문제 해결
4. 추적 가능성