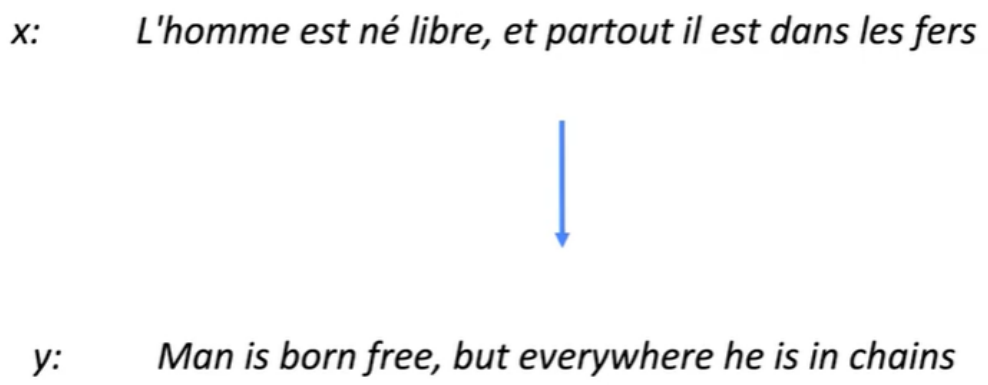
Lecture 7 - Machine Translation, Seq2Seq, Attention

> 3개의 단어에 대해 한 문장으로 표현

☝🏻 **“Machine Translation, which is a major use-case of sequence-to-sequence, which is improved by attention.”**

1. Machine Translation (기계 번역)

Pre-Neural Machine Translation



프랑스어 → 영어 (translation) input으로 들어온 x 문장에 대해서 target언어 형태로 번역하는 task를 모델이 수행하도록 하는 것을 의미한다.

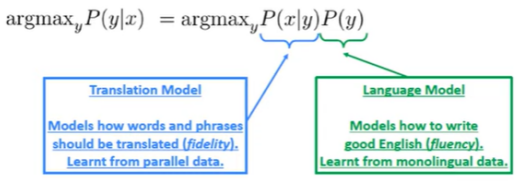
역사

- **1950s : The early history of Machine Translation**
  - 군사 목적으로 개발되기 시작(ex Russian → English)
  - 같은 뜻의 단얼르 대체하는 단순한 방식만을 사용

▼ **1990s-2010s : Statistical Machine Translation**

- Core idea : Learn a probabilistic model from data.
- 베이즈정리를 적용시켜 Translation Model + Language Model

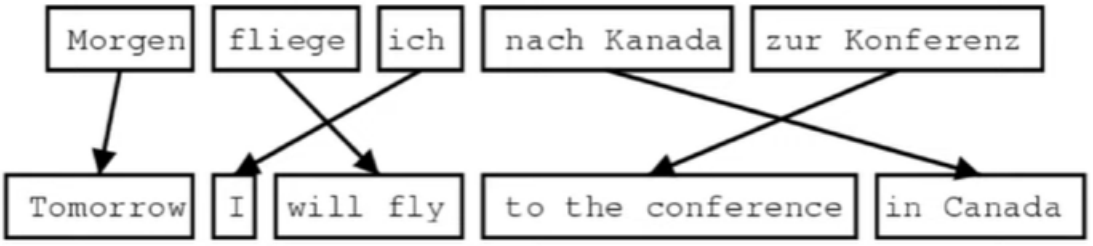
- Ex → 입력 source 문장 x에 대해 조건부확률을 최대화하는 번역 target 문장 y를 찾고자 한다.



- 기존의 테스크, 베이즈 정리를 적용시켜서 Translation Model과 Language Model로 분리
- Translation Model : 각각의 단어와 구가 어떻게 번역되어야 할지 parallel data(병렬 데이터)로부터 학습하게 된다.
- Language Model : 가장 그럴듯한 괜찮은 타겟 문장을 생성할 수 있도록 단일 데이터로부터 학습하게 된다.

Statistical Machine Translation (SMT)

- 많은 양의 parallel data가 필요로 한다.
- Source-target sentence 간의 correspondence를 단어 단위로 매핑시킬 때 **alignment 정렬 변수**를 추가적으로 사용한다.



Source, target sentence간의 1:1로 매핑되는 경우

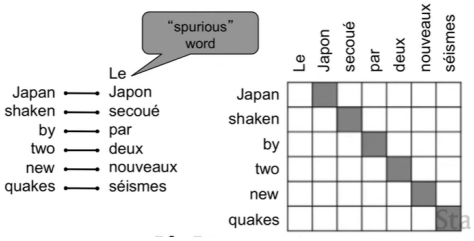
💡 하지만 다른 언어로 번역될 때 1:1로 명확하게 정의할 수 없는 경우가 훨씬 많다!! → EM 알고리즘 같은 학습 방법 사용

> EM 알고리즘 : Expectation 단계와 Maximization 단계를 각각 독립적으로 반복하는 알고리즘 (클러스터링 분야에서도 많이 사용된다.)

1:1이 아닌 복잡한 alignment 사용하는 예시

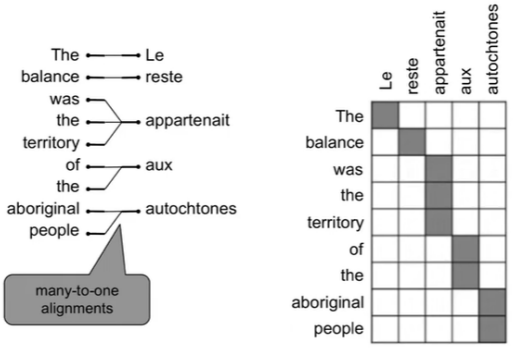
- No counterpart
- Many-to-one
- One-to-many
- Many-to-many

- **No counterpart**



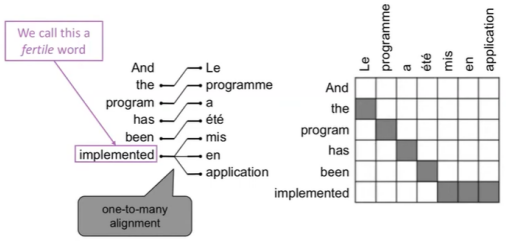
대응되는 단어가 없는 경우 존재 (가짜 word)

- **Many to one**



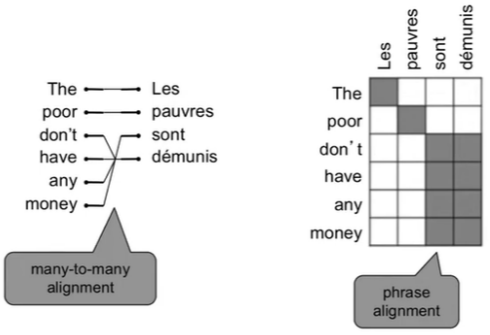
소스 문장에서 두 개 이상의 단어가 타겟 문장에서는 하나의 단어로 표현

- One to many



소스 문장에서 하나의 단어가 타겟에서 여러 단어로 번역되는 경우

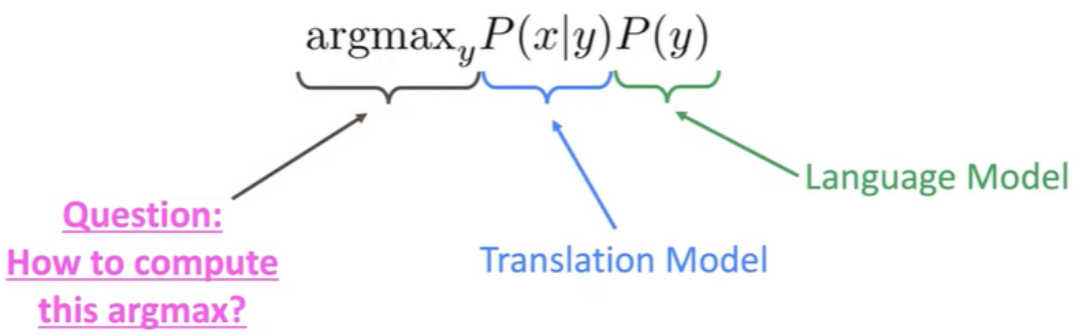
- Many to many



구 절 형태로 표현되는 관용 표현들은 다대다로 alignment가 학습되어야 한다.

→ alignment를 사용해서 번역하는 SMT 모델의 성능 떨어트린다.

Decoding for SMT



디코딩 : 학습한 translation model과 language model를 통해서 최적의 번역 문장 생성하는 과정

- 가능한 모든 y를 비교하는 방법은 너무 계산량이 많다.
- 여러 factor로 분리하고 각각의 최적해 조합으로 가장 적절한 번역문(global optimal)을 찾는 Dynamic programming 방식으로 decoding을 진행한다.

▼ 2014~ : Neural Machine Translation(NMT)

> 💡 SMT의 많은 한계점들을 극복할 수 있는 Nueral Machine Translation이다.

SMT 한계점

- 모델을 베이즈 정리로 분리하고 alignment라는 개념을 따로 정의하고 학습해야 하기 때문에, 전체 시스템을 설계하기 위해 많은 하위 시스템을 다뤄야 한다. (굉장히 복잡해짐!)
- 성능 향상을 위해서 Feature enginerring, 추가적인 resource 등이 필요로 한다. (ex phrase table)
- 또한 모델의 개발과 유지를 위해 사람의 노력이 많이 필요로 한다. (paired dataset, subcomponent 학습 등)

Neural Machine Translation (NMT, 2014~)

- 단일 end-to-end neural network → Subcomponents 별로 따로 최적화할 필요가 없고 SMT보다 성능이 뛰어나다.
- Human resource가 줄어든다. (feature engineering x, 모든 source-target pair에 대해 동일한 method 사용 가능하다.)

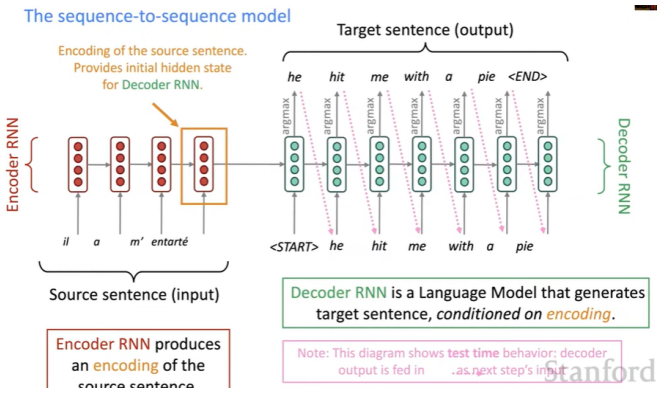
- 대부분의 딥러닝 모델처럼 해석력이 부족하다.
- Rule이나 guideline을 적용하기 쉽지 않다.

> 💡 이 신경망 구조는 2개의 RNN으로 구성된 sequence-to-sequence model(aka. seq2seq)

---

2. Seq2Seq

다음 그림과 같이 encoder와 decoder로 구성되어있다.



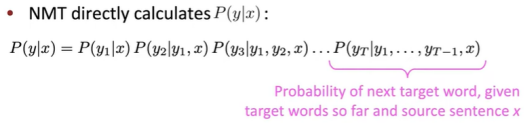
- Encoder RNN : source sentence를 encoding 한다.
- Decoder RNN은 encoding을 조건으로 target sentence를 생성하는 language model이다.
- 그림에서는 encoder의 마지막 RNN cell output이, decoder RNN에 초기 hidden state로 사용된다.
- decoder의 각 timestep 별 output이 다음 timestep의 단어를 예측하게 된다.
- Train : 학습시에는 target sentence의 timestep 별 단어가 다음 timestep의 단어를 예측하기 위한 입력으로 사용된다.
- Test : test시에는 target 문장 주어지지 않는다고 가정 → 각 step별 예측된

output이 다음 timestep의 입력으로 사용된다.

> **Machine Translation, which is a major use-case of sequence-to-sequence, …**

++ 텍스트를 입력으로 하고 다른 텍스트를 출력하는 형태의 여러 task가 seq2seq 형태로 학습 가능하다.

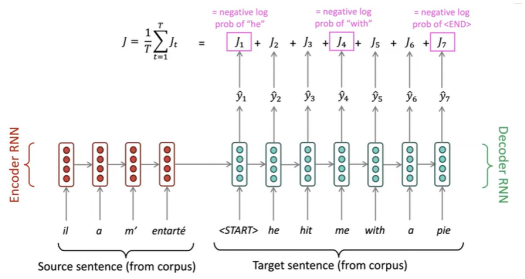
- Summarization : 긴 문장 → 핵심이 되는 짧은 문장으로 요약
- dialogue task : 다음 대화 내용 예측
- parsing : text data의 구문 분석
- Code generation ….



SMT에서 베이즈 정리로 model 분리하는 것과는 달리 조건부 확률 식 직접 계산

- $y_1, y_2 \ldots . y_t$ → 각 step별 translation된 단어
- End-to=End로 학습 가능
- NMT 학습시키기 위해서 많은 parallel corpus가 필요로 한다.

Training a Neural Machine Translation system

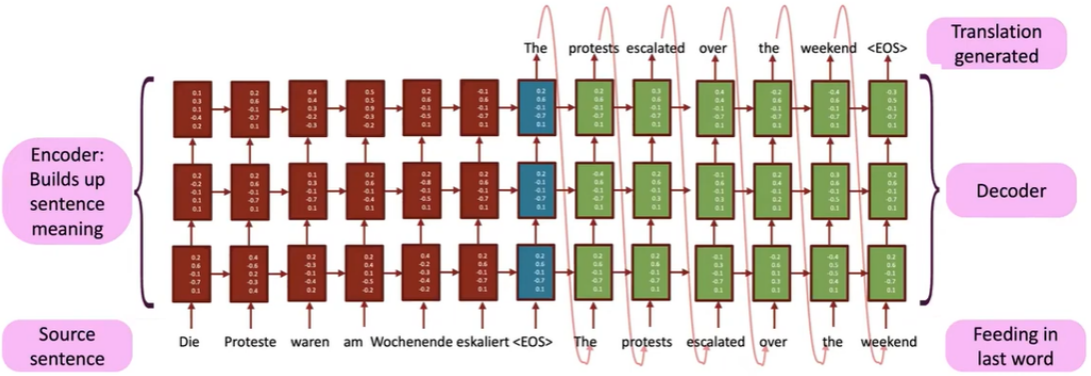


Encoder와 Decoder 과정 도식화 한 그림

- decoder의 각 step별 output는 target sentence의 한 단어 생성
- 생성된 단어, 실제 단어와의 MLE에서 음수를 붙여서 손실함수를 최소화하는 것으로 계산해서 loss로 사용되게 된다.
- NLL(negative log likelyhood) loss : [0,1] → [0, 무한대]의 범위로 매핑, 역전파 수행해서 loss를 최소화하는 문제로 정의할 수 있다.

> **위는 가장 간단한 NMT 모델이고, 성능 향상을 위해서는 여러 개의 RNN layer를 쌓는 multi-layer RNN 구조를 가진다.**

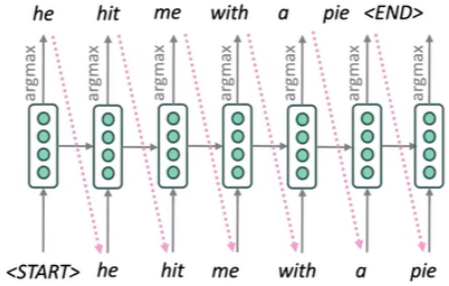
Multi-layer RNNs



multi-layer RNN → timestep의 진행과 평행한 방향으로 layer를 쌓는다.

- Multi-layer 사용 시 higher-level feature를 학습할 수 있게 한다.
- 좋은 성능을 내는 RNN 모델은 대부분 Multi-layer RNNs 구조를 사용한다.
  - 일반적으로 Encoder → 2-4 layers, Decoder → 4 layers 사용 시 가장 좋은 성능을 낸다.
  - 이후 제안된 Transformer -based(ex Bert) : 12 or 24 layers

Greedy decoding - NMT 디코딩 방식



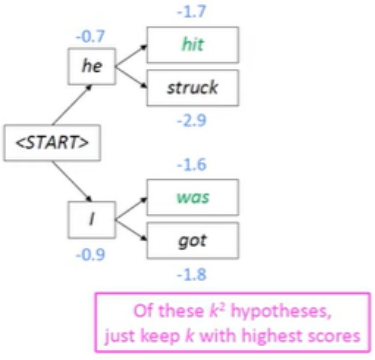
- **Exausting search 방식(생성될 수 있는 모든 target 문장 비교) 보다 효율적**
- **Step 별 argmax word를 선택하는 방식**
- **이미 선택한 단어에 대해서 번복할 수 없기 때문에 부자연스러운 문장이 생성될 수 있다. → 즉 최적해 보장 x**

- Input: *il a m'entarté* *(he hit me with a pie)*
- → *he* ____
- → *he hit* ____
- → *he hit a* ____ *(whoops! no going back now...)*

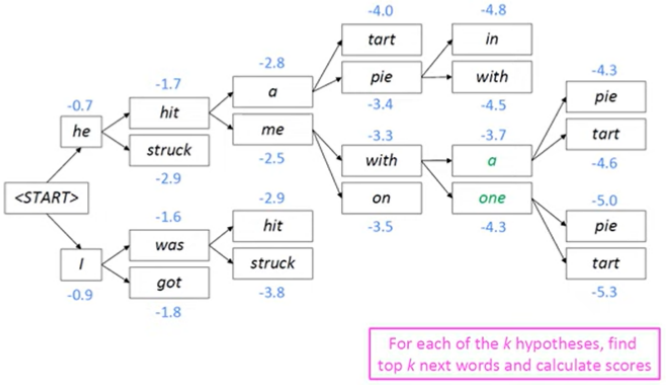
> 따라서 Greedy decoding이 아닌 Beam search decoding 방식으로 사용된다.

Beam search decoding

- k개의 best translation을 유지하면서 decoding 하는 방식이다.
  - k : beam size, usually 5-10
- 최적해 보장 x 하지만 exhaustive search보다 효율적이고, greedy decoding보다 더 자연스러운 문장 생성할 수 있음!
- k^2의 hypotheses를 비교 → k개의 best hypotheses를 선택하는 과정 반복!
- EOS 토큰 등장 or 사전에 정의된 max timestep T, 또는 n개의 완성된 문장을 종료 조건으로 사용 → 디코딩 과정 길어지는 것을 방지
- NLL loss 합을 생성된 문장 길이로 나눠서 normalize 진행 → 문장이 길어질수록 점수가 낮게 된다. (따라서 불공평하기 때문에 정규화)



k = 2인 경우 , 첫 번째 step에서 2개 단어 생성, 그 다음 step에서는 총 4개의 단어 생성해서 비교



로그 우도 값이 큰 두 개 후보 계속해서 선택하는 과정 반복

Advantage of NMT

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities.

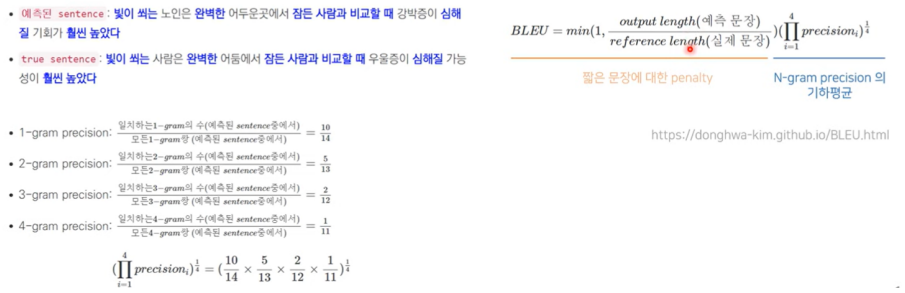
Disadvantage of NMT

- less interpretable → hard to debug
- Difficult to control (rules and guidline 적용어려움)
  - Safety concern

- A single neural network
- Requires much less human engineering effort

Evaluation for Machine Translation : BLEU

- Target sentence와 machine-written translation 비교
- N-gram precision 기반 similarity score + penalty for too short translation 사용된다.



BLEU 식 보면 penalty에 대한 term 그리고 N-gram precision 기하평균 term의 곱으로 존재

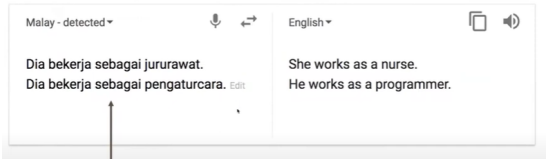
- 실제 문장 대비 예측 문장이 짧으면, 분자에 있어서 낮은 score 값(패널티) 부여된다.
- N 수의 따라서 쌍을 이뤄서 얼마나 일치한지에 대한 precision 값 구함 (단어 쌍의 수)

NMT 한계점

- 관용표현
- Bias 학습

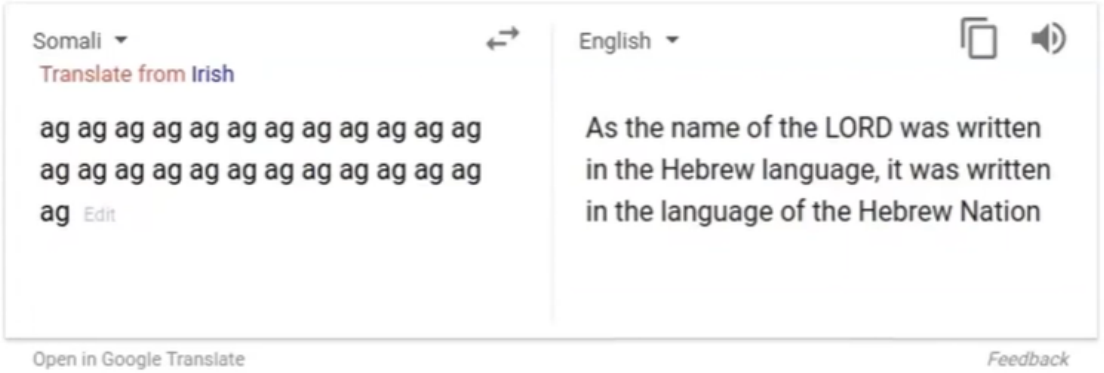
?

프린트가 종이에 끼인 것 ↔ 종이가 담긴 잼



직업에 대한 성별 bias 가지고 있는 것

- Uninterpretable

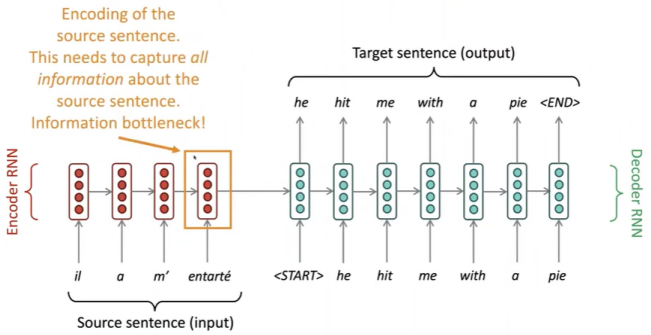


이해할 수 없는 문장에 대해서 → 임의의 문장 생성 (학습하지 않은 입력에 대해 취약함 보여줌)

---

3. Attention - NMT 모델 성능 크게 향상

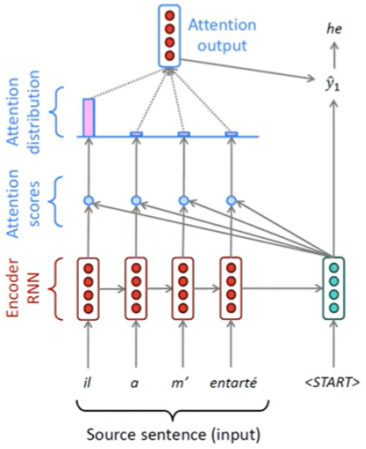
> **The bottleneck problem(병목 현상) : Source sentence의 마지막 encoding만을 입력으로 사용한다.**



Encoder와 Decoder 과정

- 마지막 인코딩에 대해서 모든 source sentence 정보를 다 담고 있어야 한다. → 하지만 대부분 모든 정보 다 담기가 어렵다.

Bottleneck 문제 해결하기 위해 제안된 Attention → Decoder의 각 step에서 source sentence의 특정 부분에 집중할 수 있도 connection 추가



Attention 사용 과정

1. **Encoder의 step별 encoding과 Decoder hidden state의 유사도를 계산해서 attention score와 attention weight(확률)를 계산**

Attention score : $\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$

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

score : 전치 및 행렬곱 형태 weight : 소프트맥스 함수 적용

2. **Attention distribution을 사용해서 encoder hidden state를 가중합하여 attention output을 계산**

Attention output : $\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$

3. **Attention output과 decoder hidden state를 concat → $\hat{y}_1$ 해당 step의 word를 생성**

Concat with attention : $[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$

Attention is great !!

- **NMT의 성능 향상 : decoder가 source sentence의 특정 영역에 집중**
- **Source sentence의 모든 embedding을 참고하는 구조를 사용하여 일반적인 Encoder-Decoder 구조의 bottleneck 문제 해결**
- **Shortcut 구조를 사용하여 Vanishing gradient 문제 줄임**
- **Alignment를 스스로 학습할 수 있고 attention probability를 통해 결과에 대한 interpretability 제공**