7. Machine Translation, Sequence-to-Sequence and Attention

1. Pre-Neural Machine Translation

Machine Translation

- 하나의 언어 x(source language)를 또 다른 언어 y로(target language)로 번역하는 작업

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

▽

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

통계적 기계 번역(Statistical machine translation, SMT)

- 1990~2010s
- 핵심 아이디어: 데이터로부터 확률 모델을 학습
  - ex. 불어(x)를 영어(y)로 번역하는 경우

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

Bayes Rule을 이용해서 위 수식을 두 개의 component로 나눌 수 있다.

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

$P(x|y)$: Translation Model, 단어나 구문이 의미가 어떻게 번역되어야하는지 병렬 데이터로부터 학습

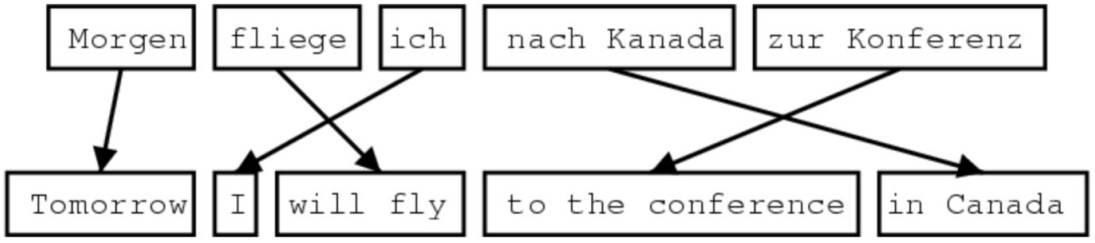
$P(y)$: Language Model, target language로 번역될 때 문장의 유창함(fluency)을 학습

SMT에서의 정렬 (alignment) 학습

- $P(x|y)$를 어떻게 학습할 수 있을까?
  → 대규모의

병렬(parallel) 데이터

→ 언어 간 문장의 대응이 어떻게 되는지를 알아야 학습을 할 수 있다!

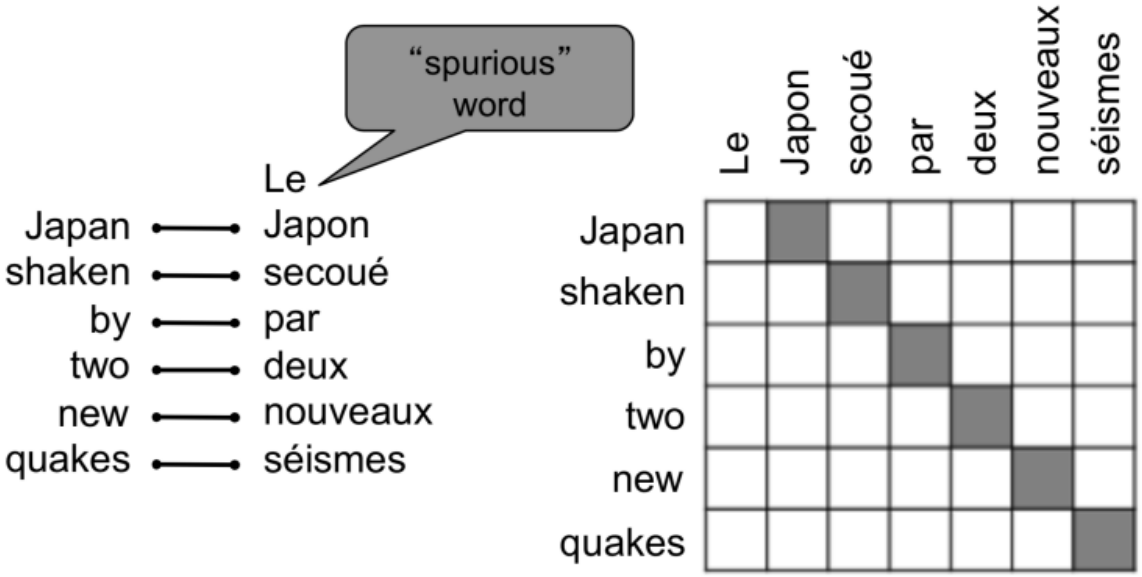


- source 문장 x와 target 문장 y 간 단어 수준에서 상응하는 것이 **정렬(alignment, a)**임
  - 이를 통해 $P(x, a|y)$를 도출
  - 정렬 a는 잠재(latent) 변수 → 병렬된 데이터 내에 명시되어 있지 x

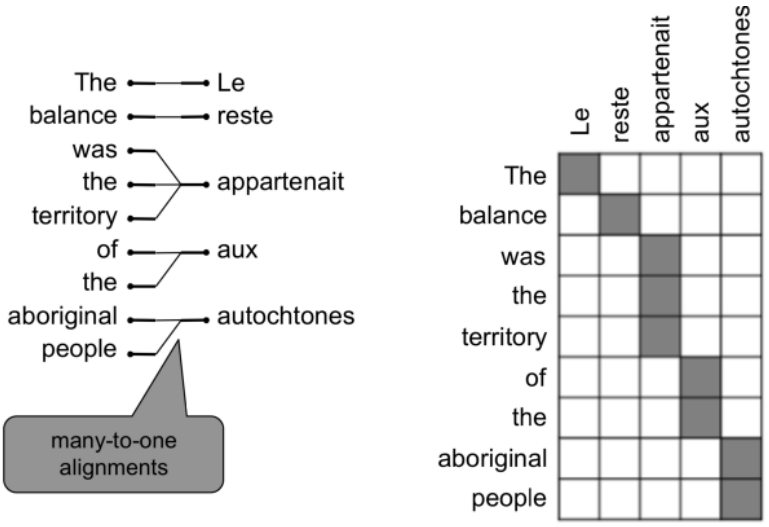
정렬(alignment)이란?

- 문장 쌍에서 특정 단어 간의 대응
- 그러나 언어 간의 특성 차이 때문에 대응이 잘 되지 않는 경우가 존재

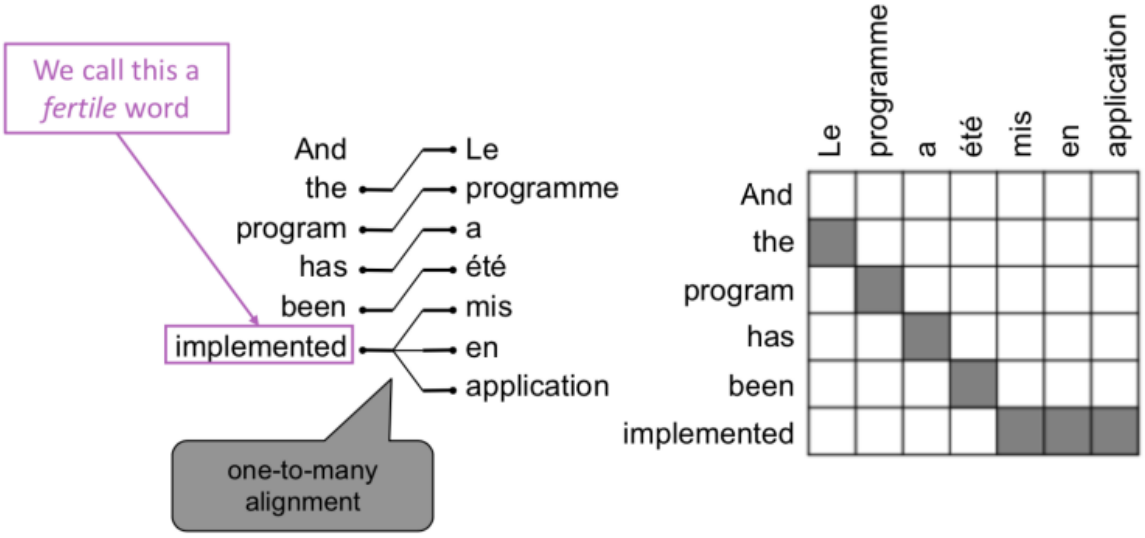
  ex. 아래의 Le처럼 대응이 없는 경우 **spurious word**라고 함



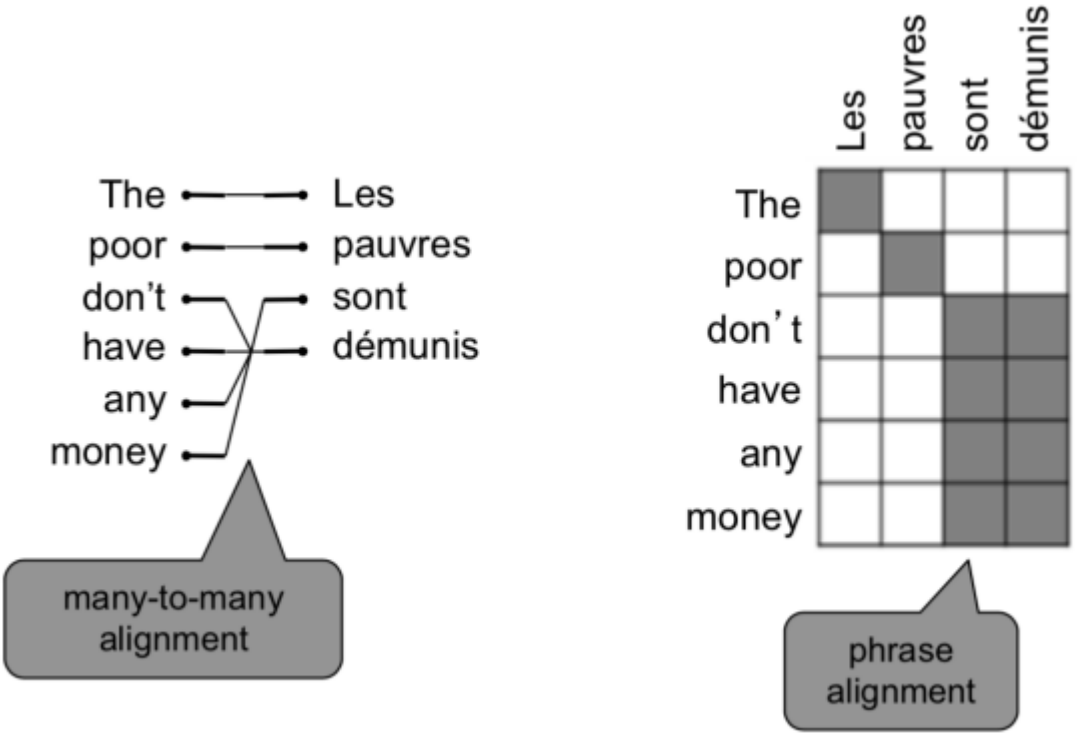
- 정렬은 다대일 대응(many-to-one), 일대다 대응(one-to-many), 다대다(many-to-many)대응이 될 수 있음



다대일 대응(many-to-one)



일대다 대응(one-to-many)



다대다 대응(many-to-many)

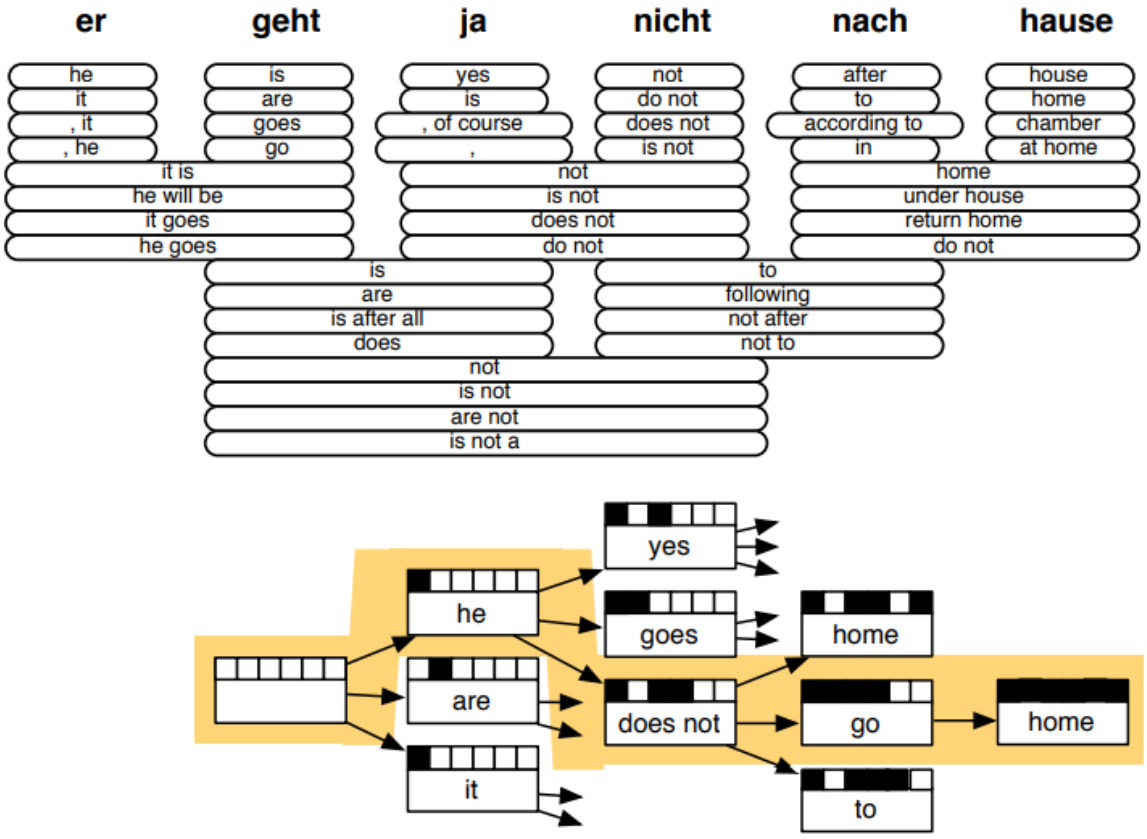
Decoding for SMT

$argmax_y P(x|y)P(y)$에서 argmax 값을 어떻게 구할까?

- 모든 경우의 수를 전부 조합하는 것은 너무 계산량을 많이 요구

  → 모델 내에 강하게 독립적이라는 가정을 적용하고, 동적 프로그래밍을 이용하여 전역 최적해를 도출

  ⇒ 이를 **decoding**이라 함



해당 decoding 방식의 예를 들면 위 그림에서 두 번째 독일 단어는 동사인데 영어에서는 올 수 없는 위치이며 재배열이 필요하다. 따라서 각 단어들의 변역될 수 있는 영단어들을 나열하여 조합해 나간다. 그래서 시작할 때 첫 단어가 he가 가장 가능성 있으므로 he를 선택하게 되고, 위의 박스는 검게 칠해진 것이 사용된 독일 단어의 위치를 표기해 준다. 이렇게 단어의 조합을 살펴보면서 가능성이 낮은 애들을 가지치기 해나가면서 가장 적절한 문장을 선택하게 된다.

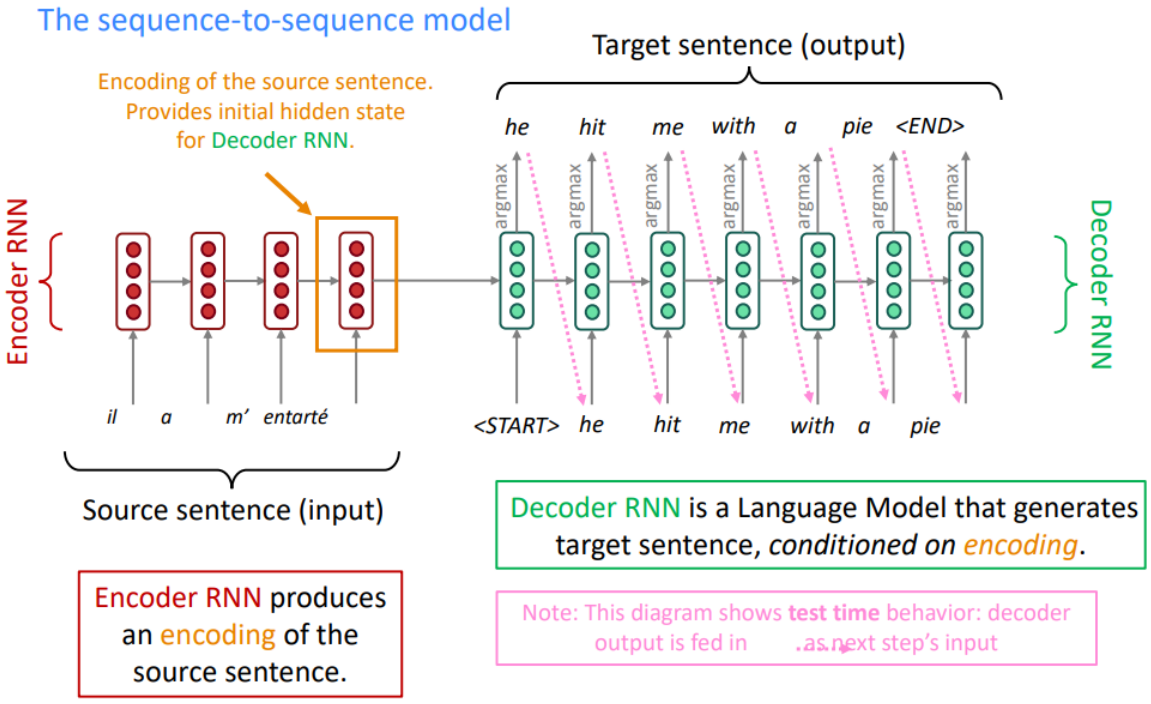
SMT의 의의 & 한계

- SMT는 큰 연구 분야이며, 최적의 시스템은 극도로 복잡함
- 시스템은 다양한 subcomponent들로 구성되었고, 많은 feature engineering을 필요로 함
- 추가 리소스 편집 및 유지 관리 필요
  - 구문 대응 표 같은..
  - 유지를 위한 human effort가 많이 들어감

2. Neural Machine Translation

신경 기계 번역(Neural Machine Translation, NMT)

- 단일 **end-to-end** (거대) 신경망을 이용하여 기계 번역을 수행하는 방법
  - 신경망 아키텍처는 sequence-to-sequence 모델 (이하 `seq2seq`)으로 불림



- source 문장을 Encoder RNN에 넣어 생성된 hidden state를 Decoder RNN에 입력으로 넣어준다.
- Decoder RNN의 경우 시작토큰 <START>와 함께 두 개의 입력을 받아 가장 적합한 단어를 추론하고, 그 단어를 또 다음 입력으로 넣어주며 문장을 생성
- `seq2seq`은 단순 기계번역(Machine Translation, MT)에만 유용한 것이 아니라 다양한 곳에 활용될 수 있음
  - 요약(긴 텍스트 → 짧은 텍스트)
  - 대화(이전 대화 → 다음 대화)
  - Parsing(입력 텍스트 → parsing 방법)
  - Code 생성(자연어 → code)

seq2seq 모델이 왜 조건부 모델인가?

- source 문장에 조건부이기 때문

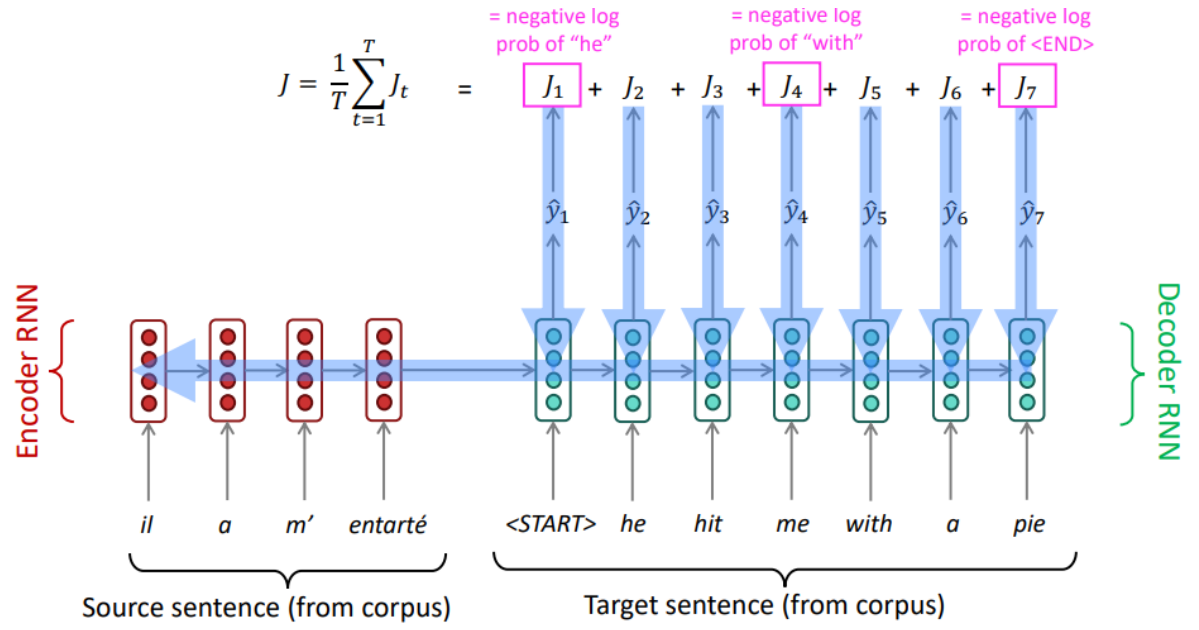
$$P(y|x) = P(y_1|x)P(y_2|y_1, x) \dots P(y_T|y_1, \dots, y_{T-1}, x)$$

- $P(y|x)$를 계산하며 source 문장 x와 target 문장일 위해 생성된 단어들을 입력으로 받아서 확률을 계산

  → 단순 언어 모델보다는 더 낮은 perplexity를 가짐

신경 기계 번역 시스템의 학습

- **큰** 병렬 말뭉치(parallel corpus)를 구하 학습



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

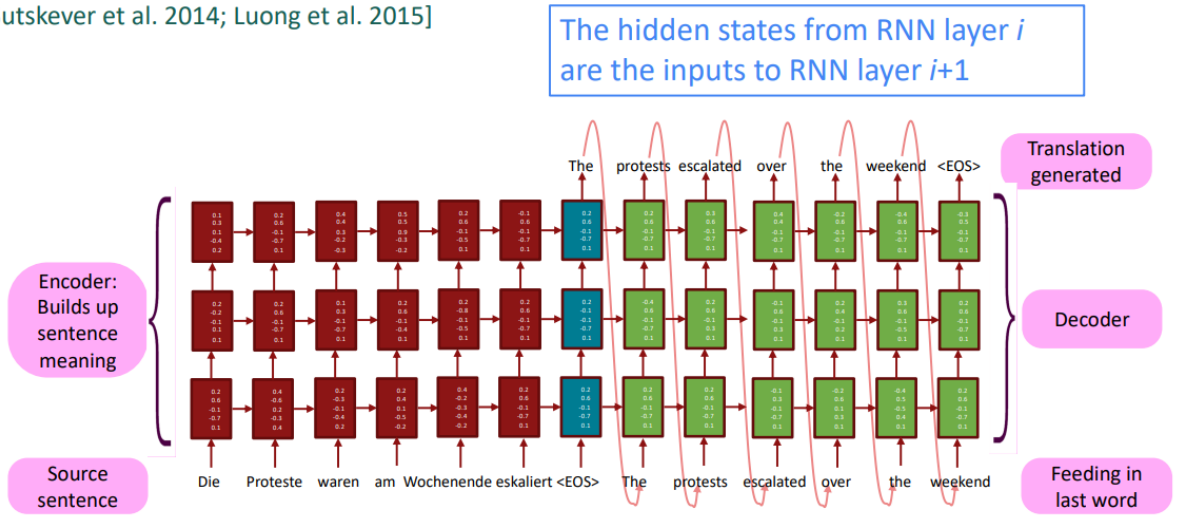
- **end-to-end 단일** 시스템으로 취급되어 최적화됨
- 시스템에서 예측한 단어와 실제 단어의 교차 엔트로피를 손실로서 구해 준 후에 teacher forcing으로 각 단어들에 대한 손실을 모두 구해줌
  - 그렇게 구한 모든 손실의 평균을 통해 역전파를 수행
  - 이 때 역전파는 decoder뿐만 아니라 encoder의 매개변수까지 업데이트!

다층 순환신경망 (Multi-layer RNNs)

- RNN은 단일 차원에서도 이미 깊은 모델임

  → 여러 timestep에 대해 순환하는 모델이기 때문
- 여러 RNN을 적용하여 또 다른 차원에 대해 깊게 적용할 수 있음
  ⇒
  multi-layer RNN
- multi-layer RNN은 더 복잡한 표현의 계산을 가능하게 해줌
  - lower RNN은 단어와 구문같은 특징인 lower-level feature를 처리
  - higher RNN은 문맥과 같은 high-level feature를 연산
- Multi-layer RNNs은 stacked RNNs로도 불림

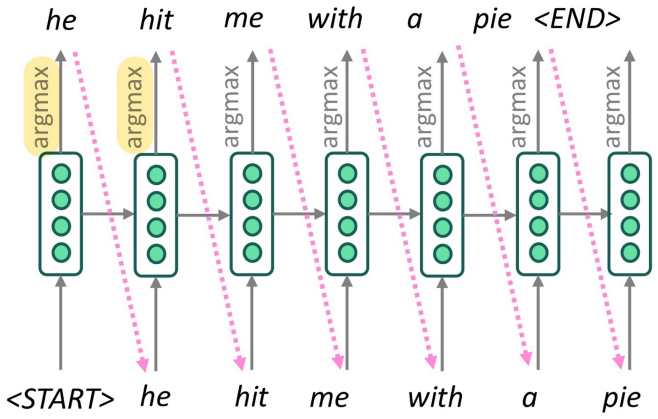
[Sutskever et al. 2014; Luong et al. 2015]



- 좋은 성능을 내는 RNN 모델들은 대부분 Multi-layer RNNs 구조를 채택함
  - Encoder with 2~4 layers, Decoder with 4 layers
  - Transformer-based: 12 or 24 layers

Greedy decoding

- 위의 decoder에서 target 문장을 생성할 때 각각의 hidden state로부터 `argmax`로 가장 확률이 높은 단어를 뽑음 ⇒ **greedy decoding**



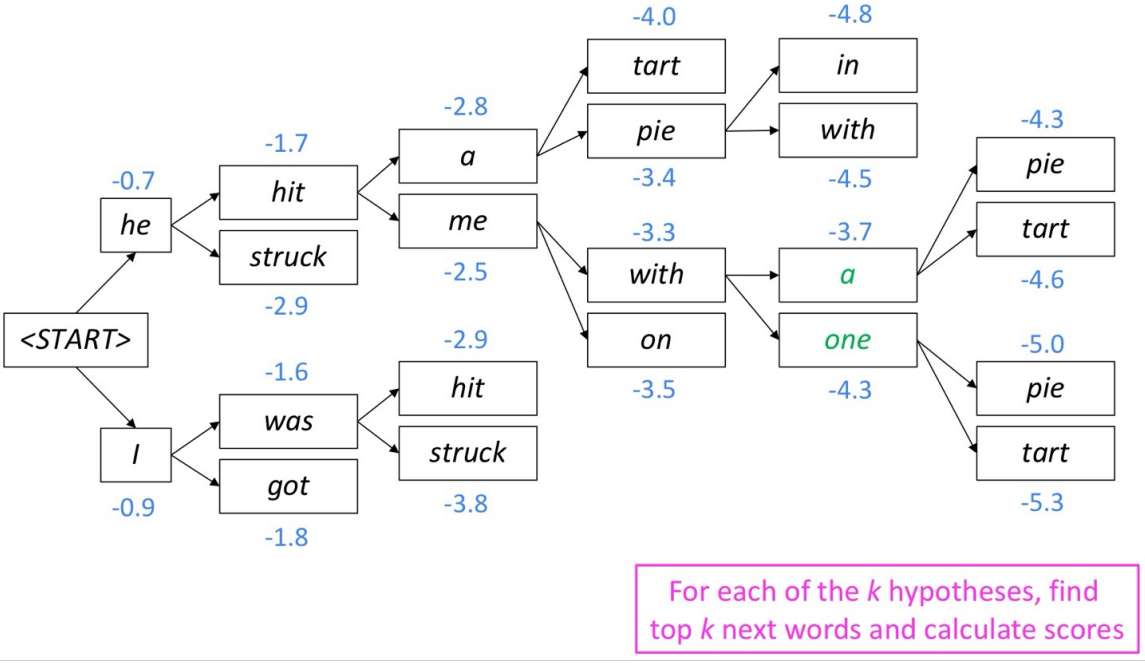
- Exhaustive search 방식보다는 효율적임
  - Step 별 argmax word를 선택하는 방식
- 문제점: 한 스탭에서 단어를 잘못 추론했을 경우 되돌아갈 수 없음

Beam search decoding

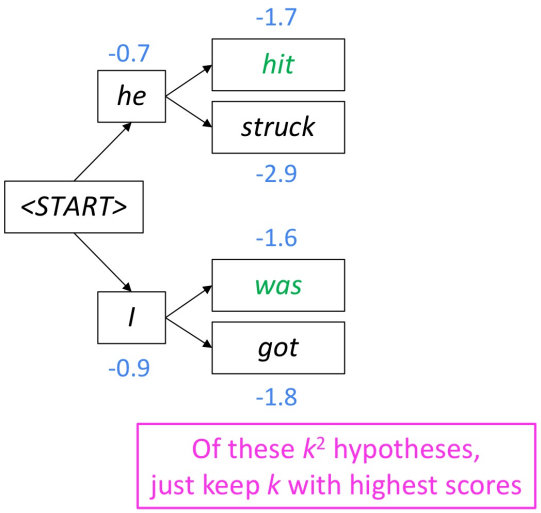
- 핵심 아이디어: 각 step에서 k개의 가장 그럴듯한 부분의 번역들을 계속 추적하는 것은 어떨까?(이를 **hypotheses**라고 함)
  - 이때 k가 beam size(일반적으로 5~10)
- hypotheses $y_1, ..., y_t$의 점수는 log 확률을 가짐

$$score(y_1, \dots, y_t) = logP_{LM}(y_1, \dots, y_t|x) = \sum_{i=1}^{t} logP_{LM}(y_i|y_1, \dots, y_{i-1}, x)$$

  - 모든 점수는 음수이고, 높을수록 좋음
  - 결과적으로 가장 높은 점수 k개를 각 단계에서 계속 추적



- 최적 해를 보장하진 않지만 exhaustive search보다 효율적이고 greedy decoding보다는 더 자연스러운 문장을 생성함
- k^2개의 hypotheses를 비교하여 k개의 best hypotheses를 선택하는 과정을 반복



- EOS 토큰의 등장이나 사전에 정의된 max timestep T 또는 n개의 완성된 문장을 종료 조건으로 함
- NLL loss 합을 생성된 문장 길이로 나눠서 normalize

NMT의 장단점

장점

1. 더 나은 성능
2. 하나의 신경망으로 최적화된 end-to-end

   → sub-component를 개별로 최적화 시킬 필요 없음
3. human engineering effort가 적게 소요됨

단점

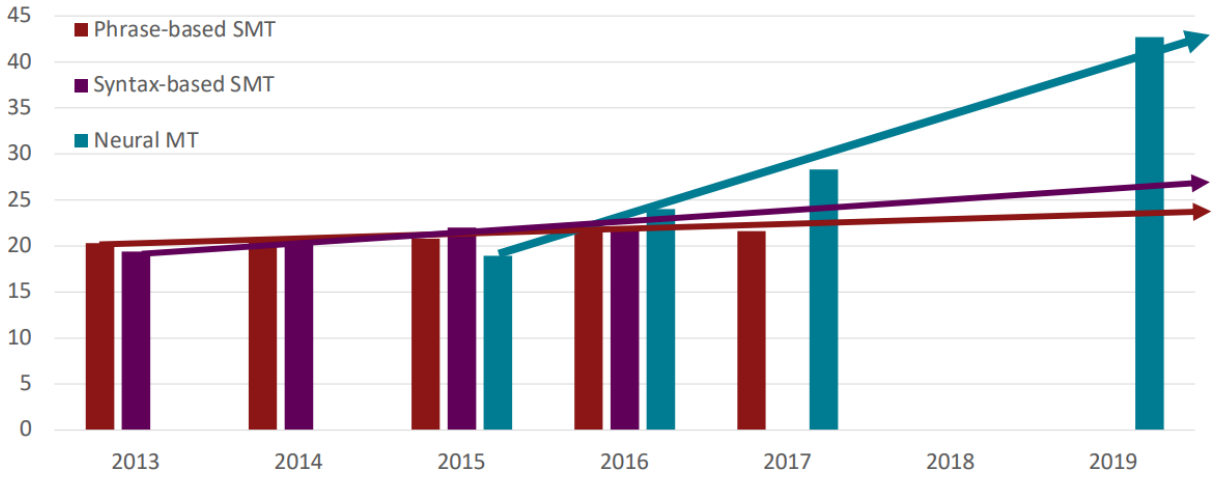
1. debuging이 어려움
2. 조절하기 힘듦

기계번역 평가 방법

- **BLEU**(**Bil**ingual **E**valuation **U**nderstudy)
  - 기계번역된 문장과 사람이 번역한 문장을 비교하여 유사도 점수를 구하는 방법
    - n-gram 정확도(일반적으로 1,2,3, 그리고 4-grams)
  - 추가적으로 짧은 문장에 대한 패널티 부여
- BLEU score는 효율적이지만 불완전함
  - 문장 번역은 다양한 방법으로 가능한데, 위 방식으로는 단순 단어만 비교하는 것이기 때문

- 좋은 번역이여도 낮은 BLEU score를 가질 수 있음

MT progress over time



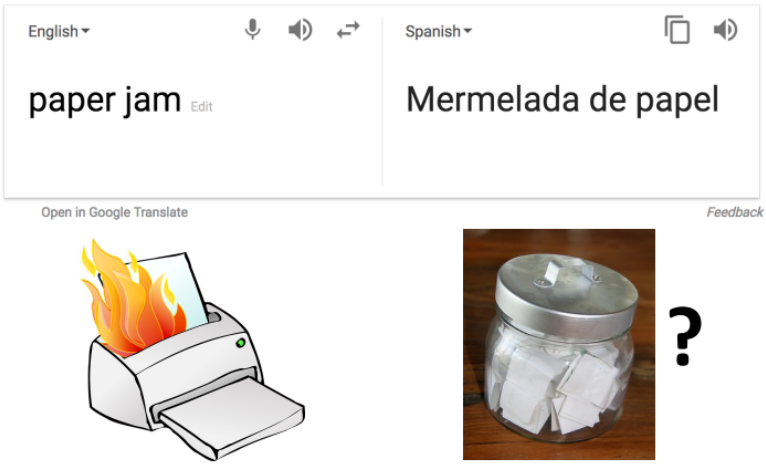
빠른 속도로 발전하고 있다.

- NMT는 2014년 이후 주력 방법으로 자리 잡았으며 2016년 구글 번역도 SMT에서 NMT로 전환되었음
- 그리고 2018년 모든 번역을 하는 기업들이 NMT를 사용하는 추세임

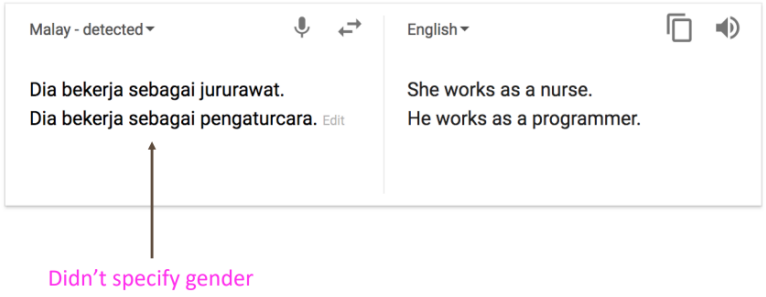
MT의 문제점은 해결되었는가,,

No!

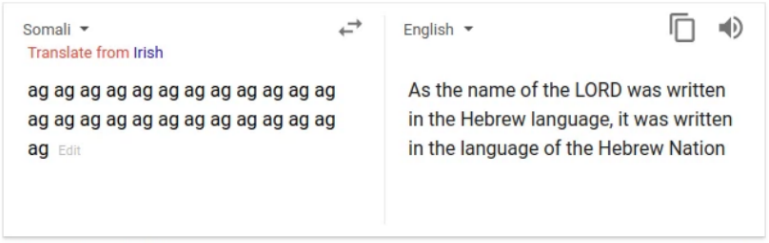
- Out-of-vocabulary words
- 학습-테스트 간 도메인 불일치
- 긴 텍스트의 문맥 유지
- 소수 언어
- 문장 의미를 정확하게 포착하지 못함
- 대명사(또는 제로 대명사) resolution 오류
- 형태학적 일치 오류



일반적인 상식에 어긋난다..



학습 데이터의 편견



해석 불가능한 시스템의 이상 행동

3. Attention

- 위에서 언급한 기계번역의 문제점을 해결하고자 **Attention** 개념을 새롭게 제안
- Intro 정도만 다루었기에, 다음 강의에서 같이 정리할랭 (ง˙∇˙)ว