#### sequence-to-sequence 학습이란?

sequence-to-sequence(Seq2Seq) 학습은 한 도메인(예: 영어 문장)에서 다른 도메인(예: 불어로 된 문장)으로 시퀀스(sequence)를 변환하는 모델 학습을 의미합니다.

"the cat sat on the mat" -> **[**Seq2Seq model] -> "le chat etait assis sur le tapis"

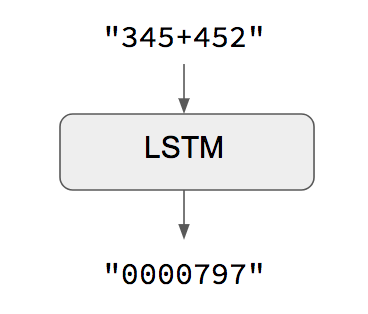
이 모델은 기계 번역 혹은 자유로운 질의응답에 사용됩니다. (자연어 질문을 주어 자연어 응답을 생성) –일반적으로, 텍스트를 생성해야 할 경우라면 언제든지 적용할 수 있습니다.

해당 작업을 다루는 여러 가지 방법이(**RNN** 혹은 **1D convnets**) 있습니다.

*이번 문서에선****RNN****을 사용하고 있습니다.*

#### 자명한(명확한) 사례 : 입력과 출력 시퀀스 길이가 같을 때

입력과 출력 시퀀스 길이가 같을 경우, 케라스 Long Short-Term Memory(LSTM)이나 GRU 계층(혹은 다수의 계층) 같은 모델들을 간단하게 구현할 수 있습니다. [예제 스크립트](https://github.com/fchollet/keras/blob/master/examples/addition_rnn.py)에선 어떻게 RNN으로 문자열로 인코딩된 숫자들에 대한 덧셈 연산을 학습할 수 있는지 보여주고 있습니다.



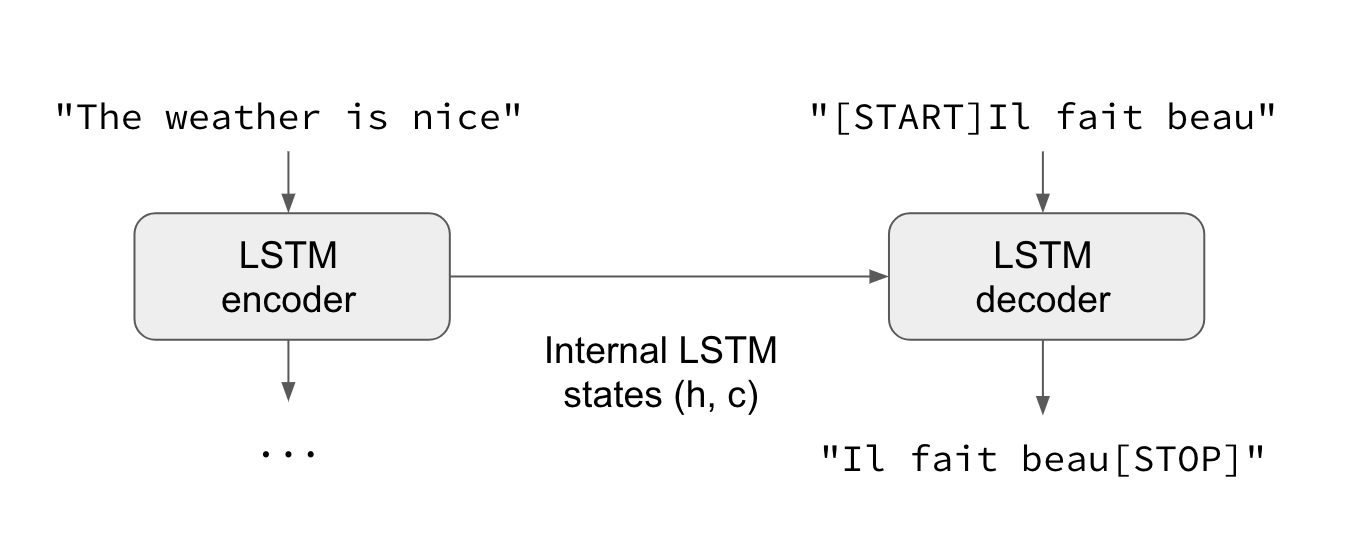
이 방법의 주의점은 주어진 input[...t]으로 target[...t]을 생성 가능하다고 가정하는 것입니다. 일부 경우(예: 숫자된 문자열 추가)에선 정상적으로 작동하지만, 대부분의 경우에는 작동하지 않습니다. 일반적으론, 목표 시퀀스를 생성하기 위해 전체 입력 시퀀스 정보가 필요합니다.

#### 일반 사례 : 표준 sequence-to-sequence

일반적으론 입력과 출력 시퀀스 길이가 다르고(예: 기계 번역) 목표 시퀀스를 예측하기 위해 전체 입력 시퀀스 정보가 필요합니다. 이를 위해 고급 설정이 필요하며, 일반적으로 “Seq2Seq models”를 언급할 때 참조합니다. 동작 방법은 하단을 참조하시면 되겠습니다.

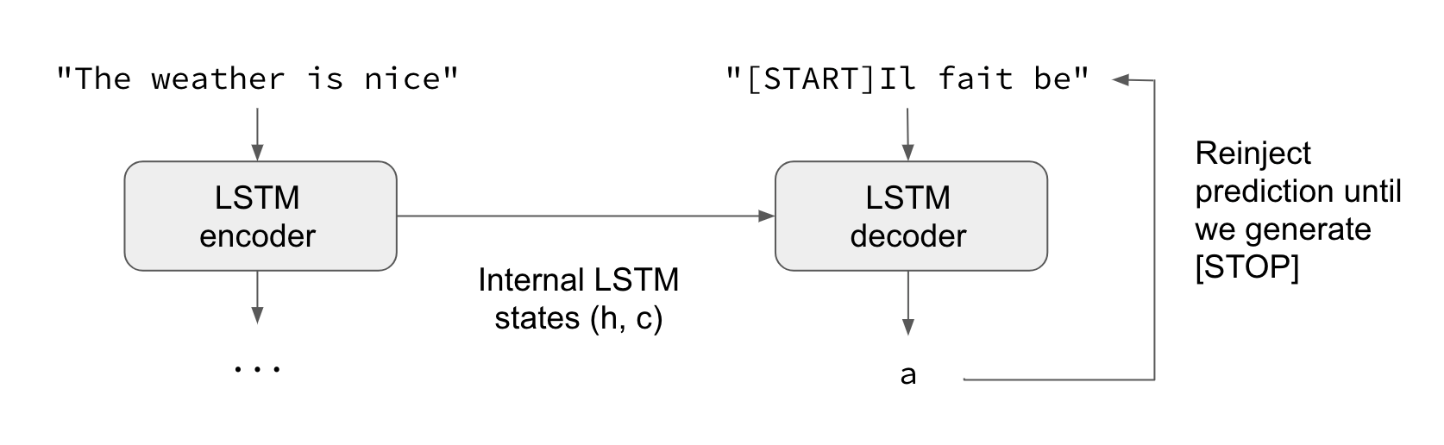
* 하나(혹은 여러 개)의 RNN 계층은 “encoder” 역할을 합니다 : 입력 시퀀스를 처리하고 자체 내부 상태를 반환합니다. 여기서, encoder RNN의 결과는 사용하지 않고 상태만 복구시킵니다. 이 상태가 다음 단계에서 decoder의 “문맥” 혹은 “조건” 역할을 합니다.
* 또 하나(혹은 여러 개)의 RNN 계층은 “decoder” 역할을 합니다 : 목표 시퀀스에서 이전 문자들에 따라 다음 문자들을 예측하도록 훈련됩니다. 상세히 말하면, 목표 시퀀스를 같은 시퀀스로 바꾸지만 후에 “teacher forcing”이라는 학습 과정인, 한 개의 time step만큼 offset\*이 되도록 훈련됩니다. 중요한 건, encoder는 encoder 상태 벡터들을 초기 상태로 사용하고 이는 decoder가 생성할 정보를 얻는 방법이기도 합니다. 사실, decoder는 주어진 target[...t]을 입력 시퀀스에 맞춰서 target[t+1...]을 생성하는 법을 학습합니다.

*offset 의 예: 문자 A의 배열이 ‘abcdef’를 가질 때, ‘c’가 A 시작점에서 2의 offset을 지님*



추론 방식(즉: 알 수 없는 입력 시퀀스를 해석하려고 할 때)에선 약간 다른 처리를 거치게 됩니다.

* 1) 입력 시퀀스를 상태 벡터들로 바꿉니다.
* 2) 크기가 1인 목표 시퀀스로 시작합니다. (시퀀스의 시작 문자에만 해당)
* 3) 상태 벡터들과 크기가 1인 목표 시퀀스를 decoder에 넣어 다음 문자에 대한 예측치를 생성합니다.
* 4) 이런 예측치들을 사용해 다음 문자의 표본을 뽑습니다.(간단하게 argmax를 사용)
* 5) 목표 시퀀스에 샘플링된 문자를 붙입니다.
* 6) 시퀀스 종료 문자를 생성하거나 끝 문자에 도달할 때까지 앞의 과정을 반복합니다.



이같은 과정은 “teacher forcing” 없이 Seq2Seq를 학습시킬 때 쓰일 수도 있습니다. (decoder의 예측치들을 decoder에 다시 기재함으로써)