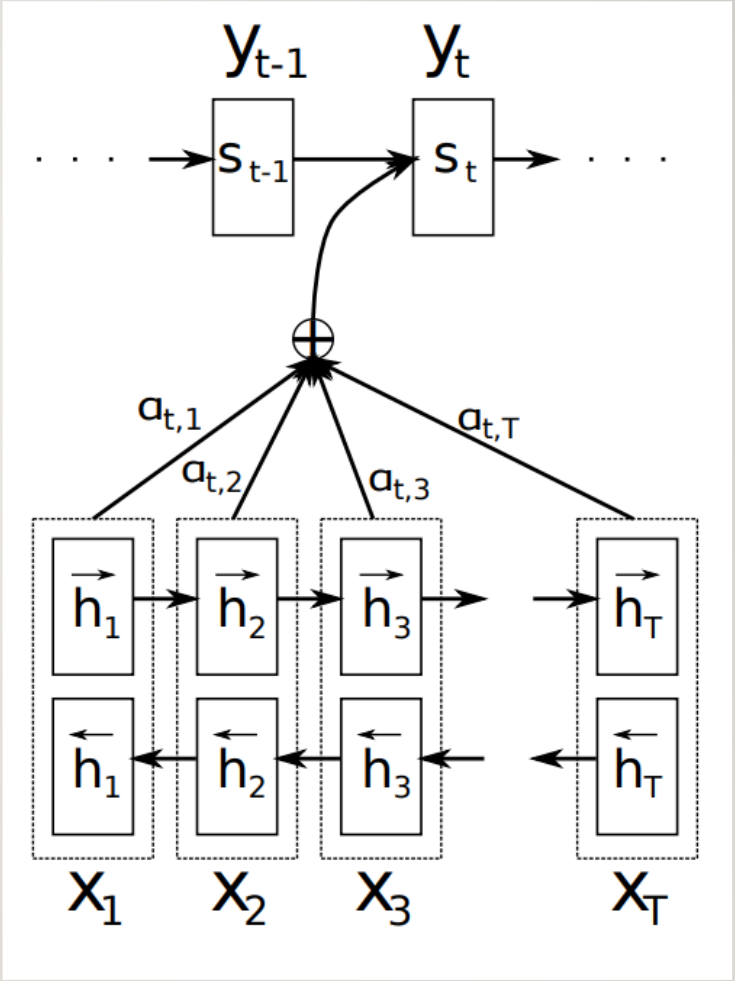
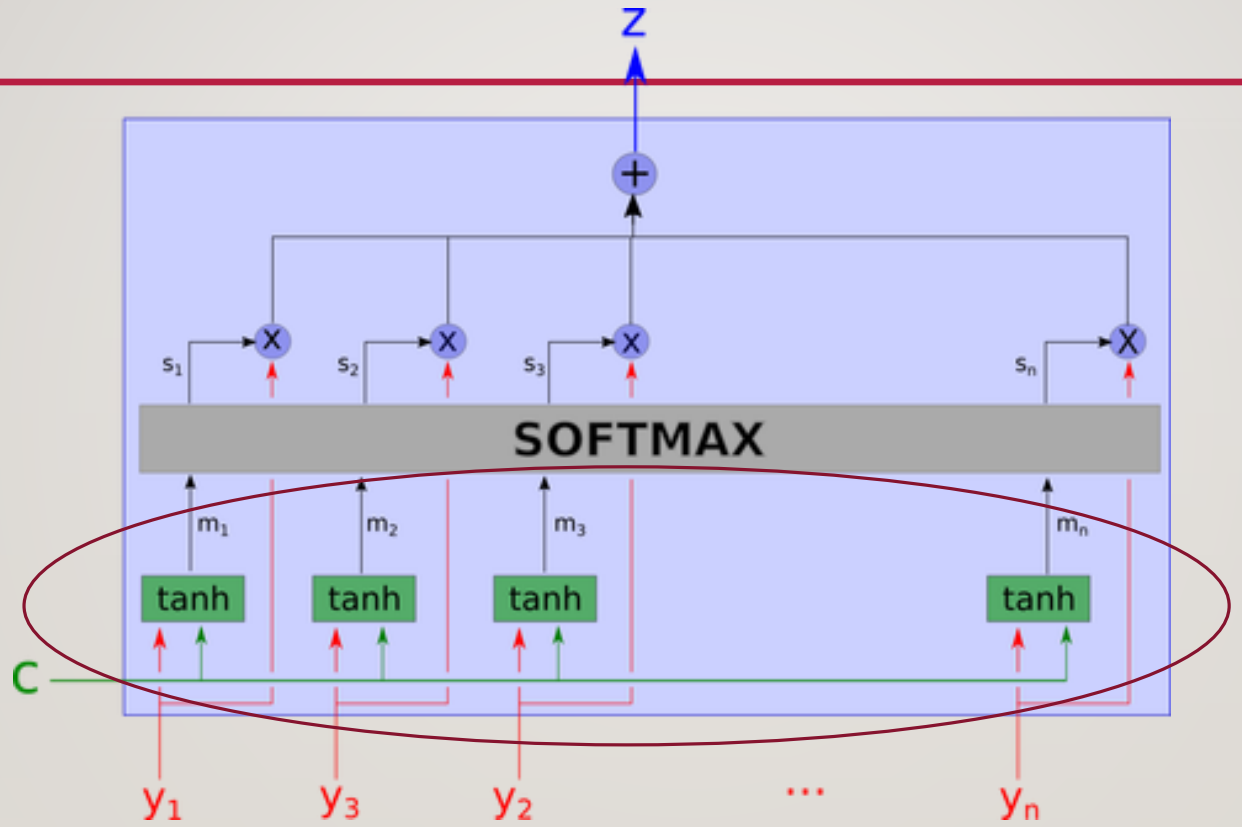
ATTENTION MODEL

ATTENTION MODEL이란?

- 딥러닝 모델이 벡터 Sequence 중에서 가장 중요한 벡터에 집중하도록 하는 모델
- State를 고려하여 가장 중요도가 높은 벡터를 중심으로 하나의 벡터로 정리하는 모델.



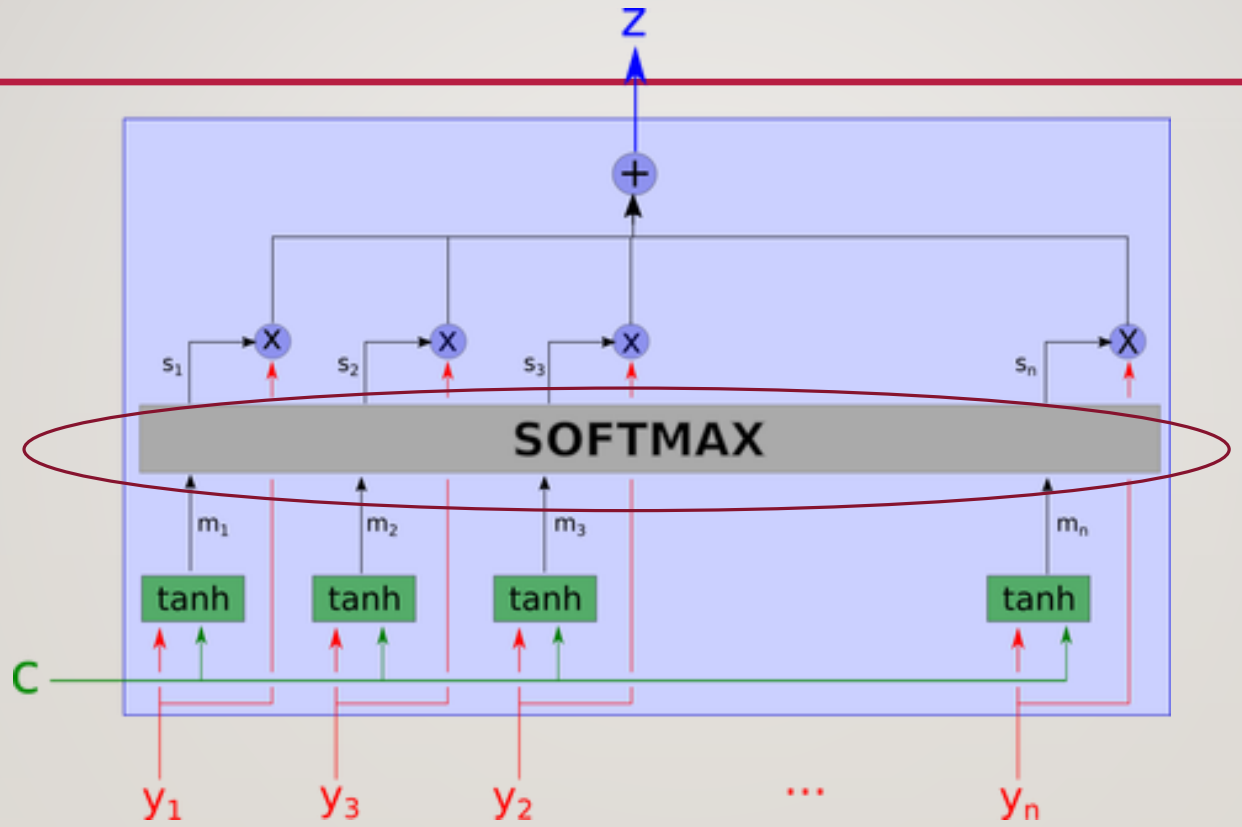
ATTENTION MODEL의 개념적 동작 (I)



Input으로 들어온 벡터들의 중요도/유사도를, 현재 State를 고려하여 구한다.

https://blog.heuritech.com/2016/01/20/attention-mechanism/

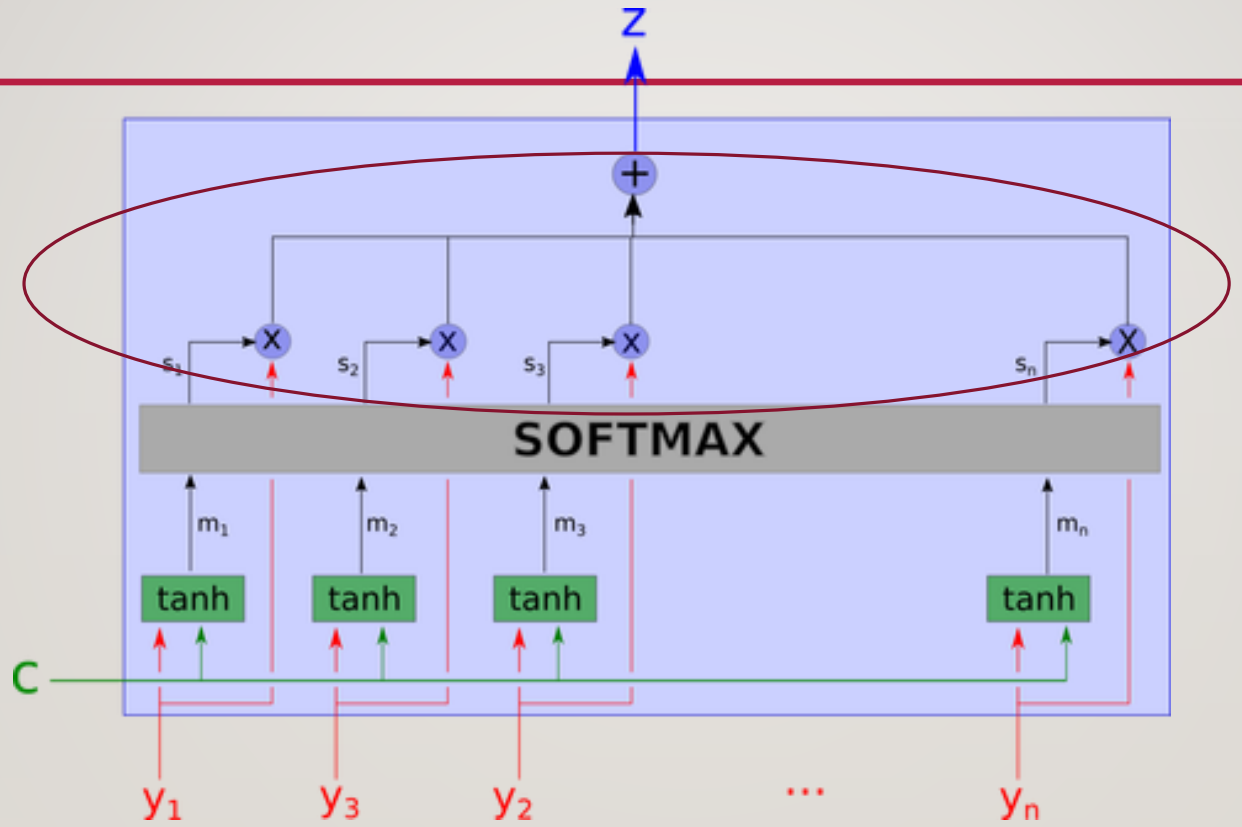
ATTENTION MODEL의 개념적 동작 (2)



각각의 중요도를, 총 합이 1.0 이 되는 상대값으로 바꾼다.

https://blog.heuritech.com/2016/01/20/attention-mechanism/

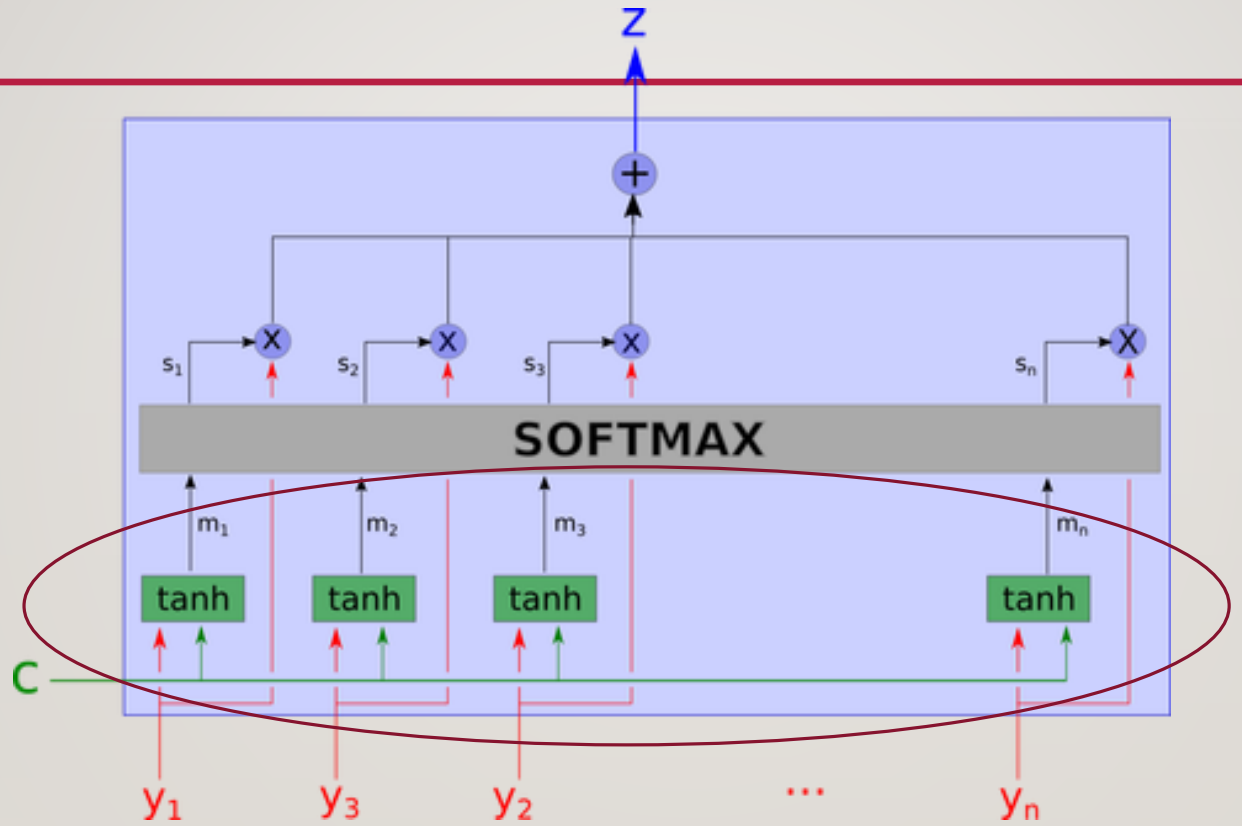
ATTENTION MODEL의 개념적 동작 (3)



상대값 중요도를 가중치로 보고, Sequence 에 있는 벡터들을 가중치합한다.

https://blog.heuritech.com/2016/01/20/attention-mechanism/

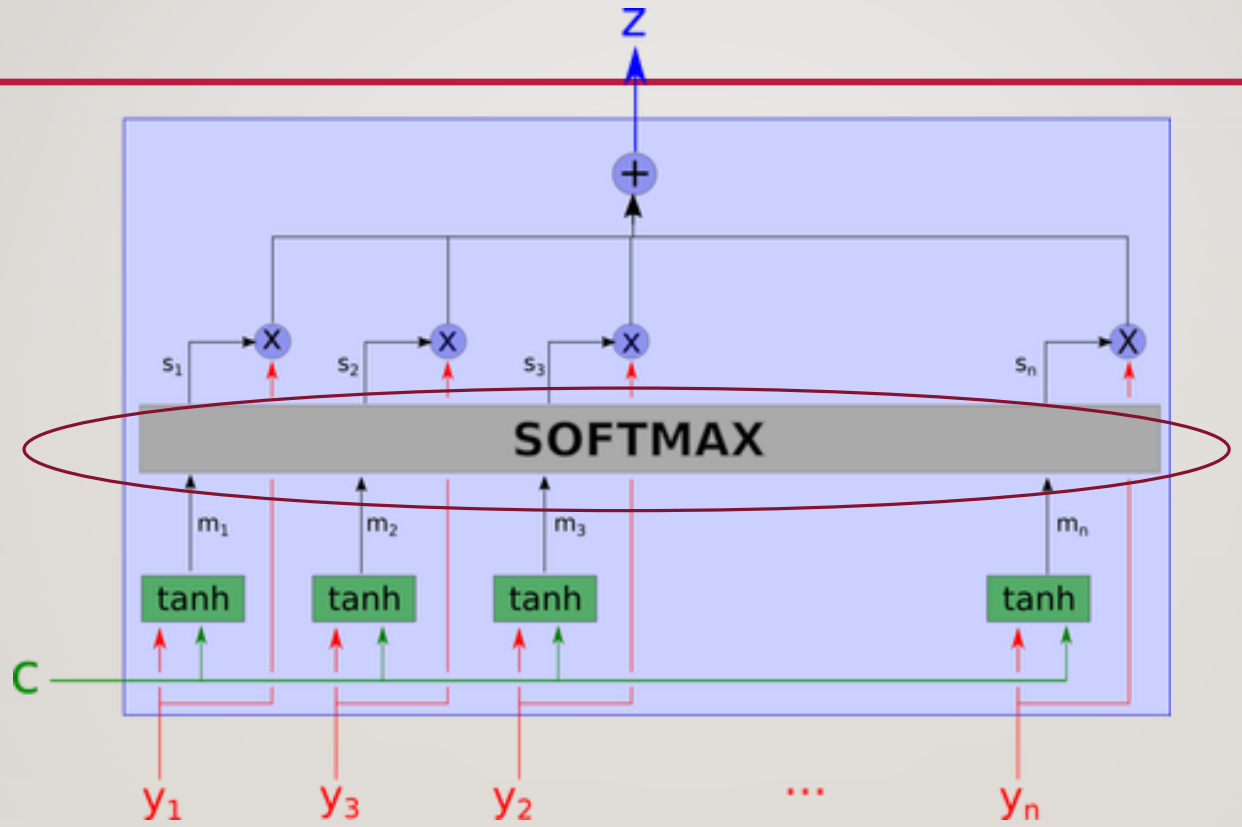
ATTENTION MODEL의 수학적 동작 (I)



State C에 W_c 행렬을 내적한 값과, 각각의 Sequence 의 벡터 y_i에 W_y행렬을 내적한 값을 더한다. 그리고 이 값을 tanh 함수에 통과킨 값을 m_i 이라고 한다.

https://blog.heuritech.com/2016/01/20/attention-mechanism/

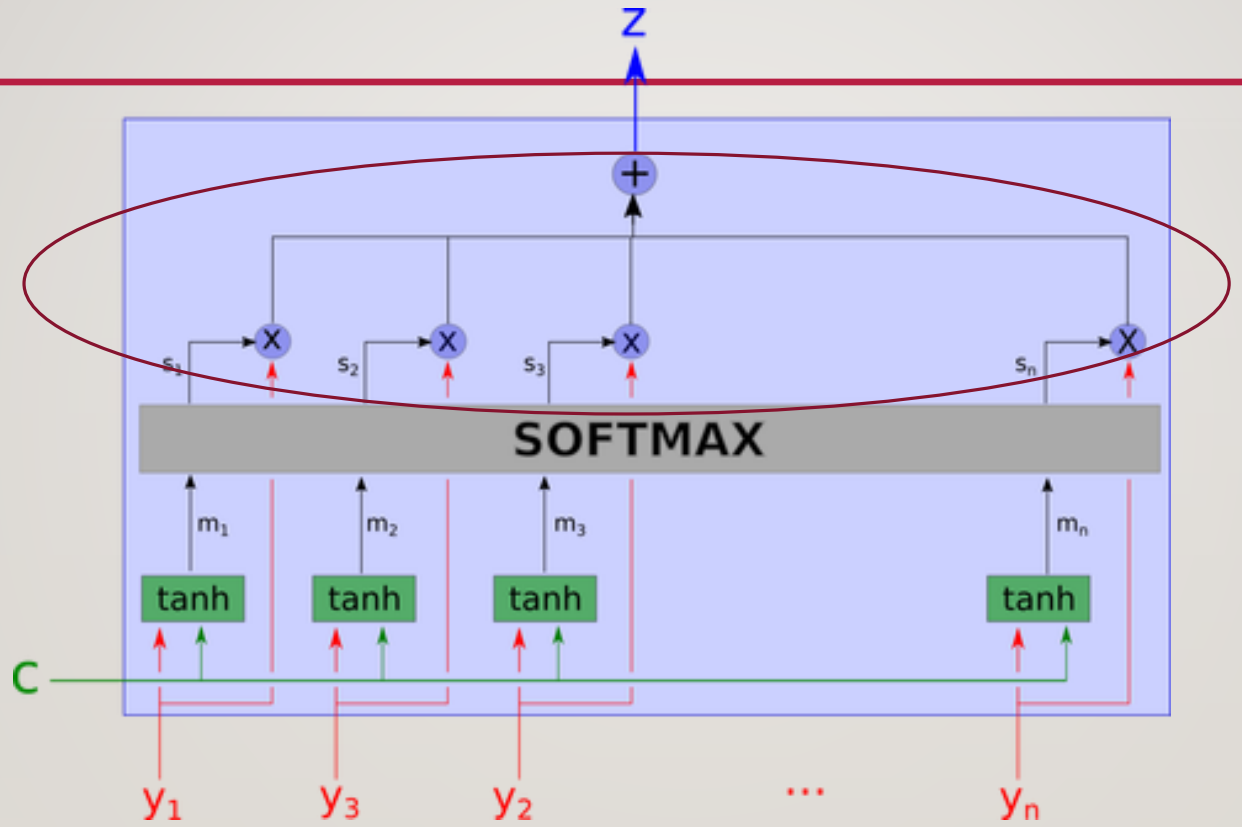
ATTENTION MODEL의 개념적 동작 (2)



m_i 값을 Softmax 함수에 통과시킨 값을 s_i라고 한다.

https://blog.heuritech.com/2016/01/20/attention-mechanism/

ATTENTION MODEL의 개념적 동작 (3)



s_i 값과 y_i 값을 내적하여 합한 것이 출력.

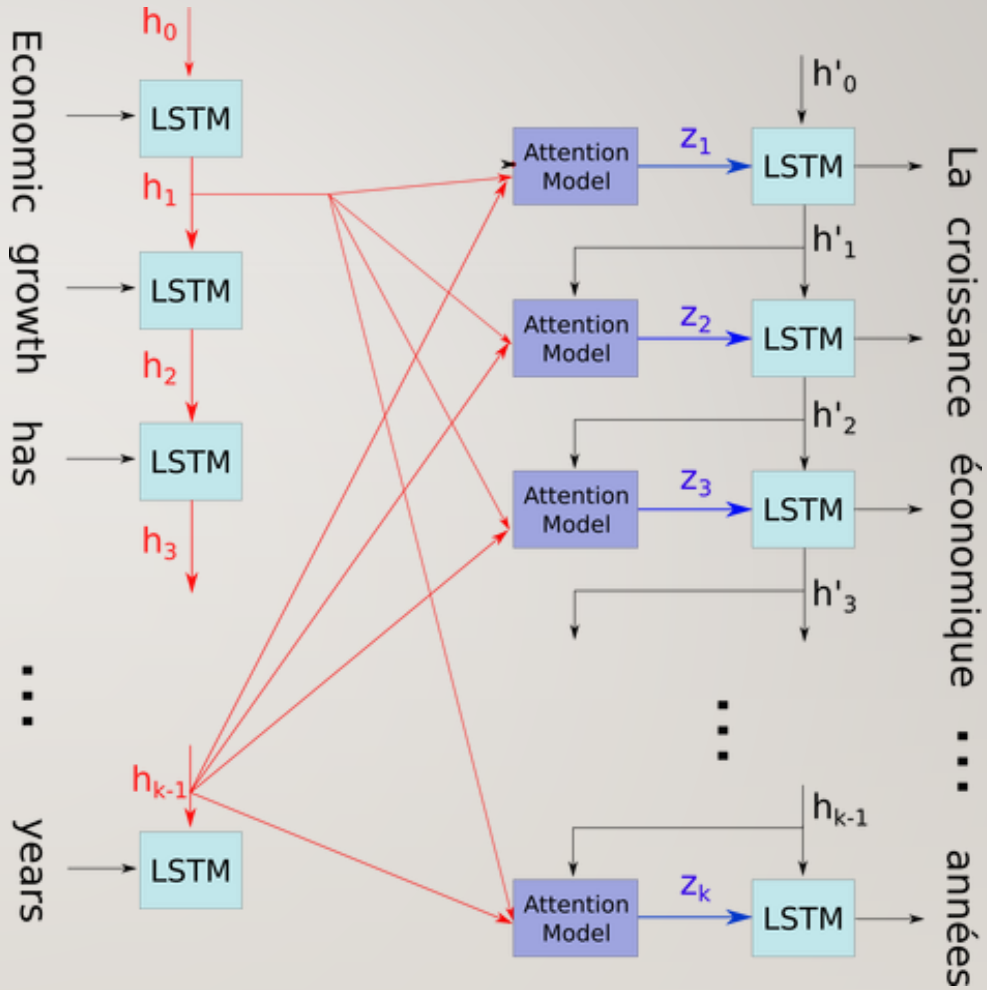
https://blog.heuritech.com/2016/01/20/attention-mechanism/

ENCODER/DECODER와 ATTENTION MODEL

- 일반적인 모델: 인코더의 모든 출력 벡터를 골고루 보고 결과를 출력한다.
- Attention Mechanism: 인코더의 출력 벡터 중 중요한 벡터에 집중한다.
  - 예) "Artificial Intelligence"를 "인공 지능"으로 번역한다고 가정한다. 이 때 모델이 '지능'을 예측할 때 'Intelligence'에 주목하게 된다.
- Decoding 과정에서 불필요한 벡터를 보지 않기 때문에 성능이 향상된다.

ENCODER/DECODER와 ATTENTION MODEL

- RNN Encoder/Decoder 를 사용할 때 Attention Model 의 Input
  - $y_1, y_2, \ldots, y_n$: 보통 이전 RNN layer 가 출력한 값의 Sequence 로 사용.
  - c: Attention Model 의 Output 을 사용하는 RNN 모델의 바로 직전 State 를 사용할 수 있다. 예를 들어 RNN 모델의 세 번째 Output 을 계산한다면, 두 번째 Output 의 hidden state 값을 사용할 수 있다.



이전 LAYER 의 출력 값이 1차원인 경우

- https://github.com/philipperemy/keras-attention-mechanism
- 수학적인 의미의 행렬을 Dense layer로 구현

```
# 이전 Layer 의 전체 값들을 Input 으로 받는다
inputs = Input(shape=(input_dims,))

# 각각의 값에 대한 중요도를 구한다.
attention_probs = Dense(input_dims, activation='softmax', name='attention_pro
bs')(inputs)

# 각각의 값 Matrix 와 중요도 Matrix 를 행렬곱한다. 즉, 입력 값들을 중요도에 따라 가
중치합한다.
attention_mul = merge([inputs, attention_probs], output_shape=input_dims, nam
e='attention_mul', mode='mul')
```

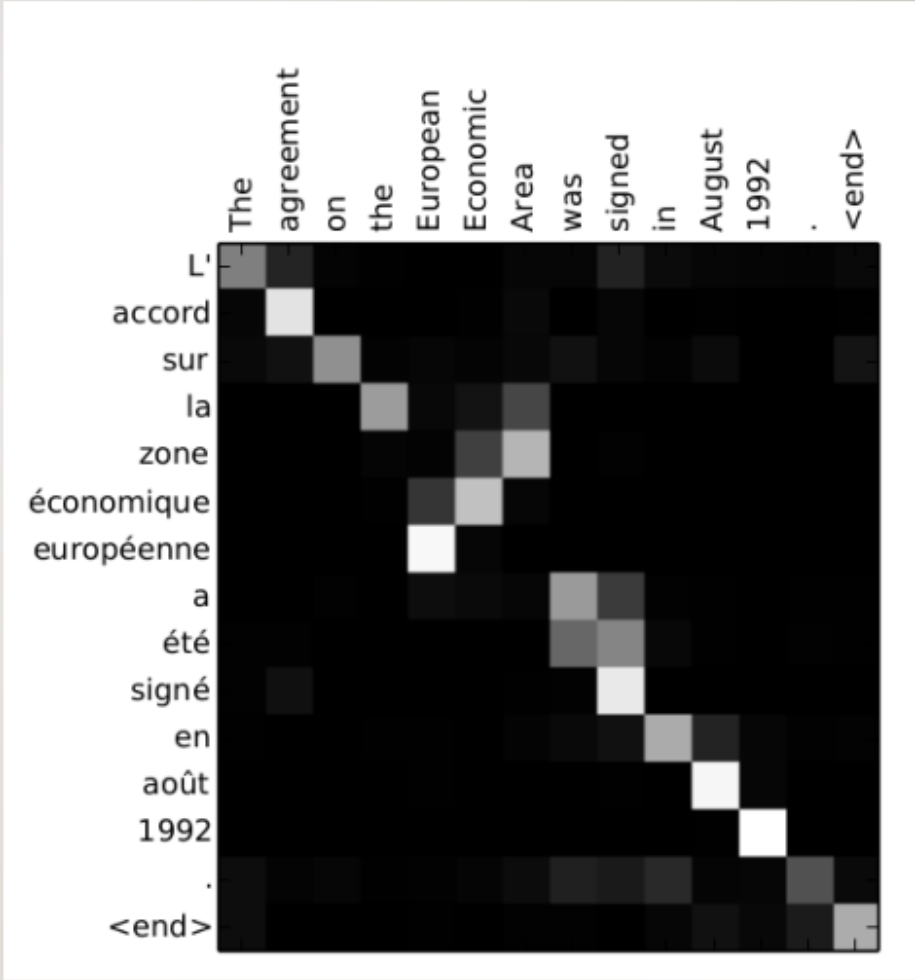
이전 LAYER의 출력이 2차원인 경우

- https://github.com/philipperemy/keras-attention-mechanism
- Input Matrix 를 Transpose 하는 이유
  - "각각의 Sequence 의 벡터 y_i 에 W_y Matrix 를 내적한 값을 더한다."를 Dense layer 로 구현하기 위해.
  - y_i 를 전체로 모은 Y Matrix 와 W_y Matrix 를 내적한다고 생각하면 된다.

```
# inputs.shape = (batch_size, time_steps, input_dim)
input_dim = int(inputs.shape[2])

 # Input Matrix 를 Transpose 한다.
a = Permute((2, 1))(inputs)
a = Reshape((input_dim, TIME_STEPS))(a)

# 각각의 값에 따라 중요도를 구한다. (LSTM 의 cell 별)
# Matrix 로 보면, 아래 Layer 의 출력은 행: sequence index, 열: input_dim
a = Dense(TIME_STEPS, activation='softmax')(a)

# 다시 Matrix 를 Transpose 해서 Input Matrix 와 차원의 의미가 같도록 수정
a_probs = Permute((2, 1), name='attention_vec')(a)
# a_probs - 행: input_dim, 열: sequence index

# Input 값들을 중요도에 따라 가중치합한다.
output_attention_mul = merge([inputs, a_probs], name='attention_mul', mode='m
ul')
```

ATTENTION MECHANISM의 활용

- RNN의 Long-Term Dependency 해결
  - Sequence가 길더라도 그 중에서 중요한 벡터에 집중할 수 있으므로
- 딥러닝 모델이 잘 학습되었는지 확인 가능
  - Input, Output에 따른 Attention 분포를 보면, 모델이 의미적으로 올바르게 학습되었는지 확인 가능



REFERENCE

- http://cs224d.stanford.edu/reports/Lichy.pdf
- https://arxiv.org/pdf/1706.03762.pdf
- https://blog.heuritech.com/2016/01/20/attention-mechanism/
- http://hugrypiggykim.com/2018/08/04/effective-approaches-to-attention-based-neural-machine-translation/
- http://docs.likejazz.com/attention/
- https://brunch.co.kr/@chris-song/9